Reliable Restricted Process Theory

Ghassemi, Fatemeh; Fokkink, Wan

published in
Fundamenta Informaticae
2019

DOI (link to publisher)
10.3233/FI-2019-1775

document version
Publisher's PDF, also known as Version of record

Fundamenta Informaticae 165 (2019) 1–41
DOI 10.3233/FI-2019-1775
IOS Press

Reliable Restricted Process Theory

Fatemeh Ghassemi*
University of Tehran
Tehran, Iran
fghassemi@ut.ac.ir

Wan Fokkink
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
w.j.fokkink@vu.nl

Abstract. Malfunctions of a mobile ad hoc network (MANET) protocol caused by a conceptual mistake in the protocol design, rather than unreliable communication, can often be detected only by considering communication among the nodes in the network to be reliable. In Restricted Broadcast Process Theory, which was developed for the specification and verification of MANET protocols, the communication operator is lossy. Replacing unreliable with reliable communication invalidates existing results for this process theory. We examine the effects of this adaptation on the semantics of the framework with regard to the non-blocking property of communication in MANETs, the notion of behavioral equivalence relation and its axiomatization. To utilize our complete axiomatization for analyzing the correctness of protocols at the syntactic level, we introduce a precongruence relation which abstracts away from a sequence of multi-hop communications, leading to an application-level action preconditioned by a multi-hop constraint over the topology. We illustrate the applicability of our framework through a simple routing protocol. To prove its correctness, we introduce a novel proof process, based on our precongruence relation.

Keywords: Mobile ad hoc network, restricted broadcast, process algebra, behavioral congruence, refinement.

*Address for correspondence: University of Tehran, Tehran, Iran.

Received March 2017; revised October 2018.

1. Introduction

The applicability of wireless communication is growing rapidly in areas like home networks and satellite transmissions, due to their broadcasting nature. Mobile ad hoc networks (MANETs) consist of several portable hosts with no pre-existing infrastructure, such as routers in wired networks or access points in managed (infrastructure) wireless networks. The design of MANET protocols is complicated, because due to mobility of nodes the topology of communication links is dynamic. Important MANET protocols such as the Ad hoc On Demand Distance Vector (AODV) routing protocol [43] contained flaws in their original design and have been revised accordingly. Formal methods can be applied in the early phases of the protocol development to analyze and capture conceptual errors before their implementation. For instance, some errors in the design of AODV were found in [6, 40, 33, 51] using formal techniques.

There are numerous applications of existing formal frameworks such as SPIN [46, 49, 6] and UPPAAL [49, 50, 26, 36, 11, 15] for the analysis of MANET protocols. Lack of support for compositional modeling and arbitrary topology changes motivates developing a new approach, tailored to the domain of MANETs, with a primitive for local broadcast and supporting the verification of MANET protocols against changes of the underlying topology. The tailored formal modeling framework should provide some form of wireless communication which varies at the different layers of the Open Systems Interconnection (OSI) model: physical, data link, network, transport, session, presentation, and application. For instance, the data link layer is responsible for transferring data across the physical link and handling conflicts due to simultaneous accesses to the shared media. In contrast, communication at the network layer provides point-to-point communication between two nodes that are not directly connected through appropriate routing of messages by using the communication service of the data link layer. Most frameworks for the formal analysis of MANET protocols, such as [41, 24, 25, 37, 42, 47, 27, 34, 16, 51], focus on protocols above the data link layer; hence they support the core services of this layer, which means that local broadcast is the primitive means of communication. Wireless communication at this layer is *non-blocking*, i.e., the sender broadcasts irrespective of the readiness of its receivers, and is *asynchronous*, i.e., received packets are buffered at the receiver. The data link layer of a node processes the packet if it is an intended destination. While a node is busy processing a message, it can still receive messages, buffer them and process them later. However, if two different nodes broadcast simultaneously with a common node in their range, the latter node cannot receive both messages and drops one of them, which is called the hidden node problem. We say that wireless communication is *reliable* if the intended receivers successfully receive the packet. In other words, message delivery is guaranteed to all connected neighbors.

Although lossy communication is an integral part of MANETs, mimicking it faithfully in a formal framework can hamper the formal analysis of MANET protocols. To obtain a deeper understanding of a malfunctioning of such a protocol due to a conceptual mistakes in its design rather than unreliable communication, it may be helpful to consider communication reliable, meaning that the possibility of the hidden node problem is omitted from the framework [33, 16]. Therefore we introduced the process algebra *Reliable Restricted Broadcast Process Theory (RRBPT)* in [18], to perform model checking of MANET protocols in a setting where communication is reliable. It is a variant of Restricted Broadcast Process Theory *(RBPT)* that we introduced previously in [19] for the modeling and analysis of

protocols above the data link layer. The underlying semantic model of *RBPT*, a so-called constrained labeled transition system (CLTS), implicitly considers mobility of nodes with the novel notion of a network constraint, which abstractly defines a set of topologies: those satisfying the given connectivity constraints. The transitions of a CLTS are annotated with appropriate network constraints to restrict the behavior to MANETs with a topology of the specified ones. *RBPT* was extended with a set of auxiliary operators to reason about MANETs by equational reasoning, so-called Computed Network Process Theory (*CNT*) [21]. We provided a sound and complete axiomatization for *CNT* terms with finite-state behaviors, modulo so-called rooted branching computed network bisimilarity. This axiomatization enables linearization of processes at the syntactic level to take advantage of symbolic verification [31, 17], especially when the network is composed of similar nodes [32, 22].

Somewhat surprisingly, all these results do not carry over in a straightforward fashion from *RBPT* to *RRBPT*. To put the model checking approach presented in [18] on a firm basis, the current paper develops the formal foundations for *RRBPT* and modifies the core of *CNT*. In a lossy setting, the non-blocking property of local broadcast communication is an immediate consequence of the rule Par and its counterpart for the parallel composition: $\frac{t_1 \xrightarrow{a} t_1'}{t_1 \parallel t_2 \xrightarrow{a} t_1' \parallel t_2}$, which expresses that if a node is not ready to participate in a communication, then we can assume that either it was disconnected from the sender or it was connected but has lost the message. However, in the reliable setting, to guarantee the non-blocking property, nodes should always be input-enabled. *RRBPT* provides a sensing operator which allows to change the control flow of a process depending on the status of node connectivity with other nodes. The input-enabledness feature is ensured through the *RRBPT* operational rules, where the main difference between *RRBPT* and *RBPT* is: in *RRBPT*, nodes lose a communication only when they are disconnected and are always input-enabled. We recap challenges of bringing input-enabledness feature in the semantics of *RRBPT* in the presence of the sensing operator. Furthermore, the behavioral equivalence relation of *CNT* setting is not a congruence with respect to parallel composition anymore. To support the desired distinguishing power, we provide a new bisimulation relation which guarantees the congruence property for MANETs. *RRBPT* can be extended in the same way as *RBPT* with computed network terms and the auxiliary operators *left merge* ($\between$) and *communication merge* ($|$) to provide a sound and complete axiomatization for the parallel composition. However, the input-enabledness feature and the new sensing operator require new auxiliary operators to assist their axiomatization. To this aim, we discuss the appropriate axioms of *RRBPT*. We utilize our axioms to analyze the correctness of protocols at the syntactic level. To this aim, we facilitate the specification of the protocol behaviors preconditioned to multihop constraints and then introduce a new notion of refinement among protocol implementations and their specifications. Such a relation abstracts away from a sequence of multi-hop communications, leading to an application-level action preconditioned by a multi-hop constraint over the topology. Therefore, the correctness of a protocol (with a finite-state behavior) is accomplished by proving that the implementation rewritten into a recursive specification by our axiomatization (made of only dynamic operators) refines the specification. We demonstrate the applicability of our framework by analyzing and proving the correctness of a simple routing protocol inspired by the AODV protocol.

This paper is organized as follows. Sections 2 and 4 introduce our semantic model and explain how it is helpful in giving semantics to reliable communication. Section 3 introduces the syntax of

RRBPT. Sections 5 and 6 provide the appropriate notion of behavioral equivalence and axioms in the reliable setting, respectively. We demonstrate the applicability of our new framework by analyzing a simple routing protocol in Section 7. We review and compare the related process algebraic frameworks in depth in Section 8 before concluding the paper.

2. Constrained labeled transition systems

Let Loc denote a set of network addresses, ranged over by ℓ. Viewing a network topology as a directed graph, it can be defined as $\gamma : Loc \to I\!P(Loc)$, where $\gamma(A)$ expresses the set of nodes that are directly connected to A, and hence, can receive messages from A. A network constraint $\mathcal{C}$ is a set of connectivity pairs $\rightsquigarrow$: $Loc \times Loc$ and disconnectivity pairs $\not\rightsquigarrow$: $Loc \times Loc$. In this setting, non-existence of (dis)connectivity information between two addresses implies lack of information about this link (which can e.g. be helpful when the link has no effect on the evolution of the network). For instance, $B \rightsquigarrow A$ denotes that A is connected to B directly and consequently A can receive data sent by B as before, while $B \not\rightsquigarrow A$ denotes that A is not connected to B directly and consequently cannot receive any message from B. The direction of an arrow shows the direction of information flow. We write $\{B \rightsquigarrow A, C,\ B \not\rightsquigarrow D, E\}$ instead of $\{B \rightsquigarrow A,\ B \rightsquigarrow C,\ B \not\rightsquigarrow D,\ B \not\rightsquigarrow E\}$. The set Loc is extended with the unknown address ? to represent the address of a node which is still not known or concealed from an external observer. For instance, the leader address of a node can be initialized to this value. Furthermore, to define the semantics of communicating nodes in terms of restrictions over the topology in a compositional way, the semantics of receive actions can be defined through an unknown sender, which will be replaced by a known address when the receive actions are composed with the corresponding send action at a specific node (see Section 4).

A network constraint $\mathcal{C}$ is said to be *well-formed* if $\forall \ell, \ell' \in Loc\ (\ell \rightsquigarrow \ell' \notin \mathcal{C} \vee \ell \not\rightsquigarrow \ell' \notin \mathcal{C})$. Let $\mathbb{C}^v(Loc)$ denote the set of well-formed network constraints that can be defined over the network addresses in Loc. We define an ordering on network constraints. We say that $\mathcal{C}_1 \preccurlyeq \mathcal{C}_2$ iff $\mathcal{C}_2 \subseteq \mathcal{C}_1$ or $\exists\, \ell \in Loc\, (\mathcal{C}_2[\ell/?] \subseteq \mathcal{C}_1)$, where $d[d_1/d_2]$ denotes the substitution of d_1 for d_2 in d; this can be extended to processes. E. g., $\{B \rightsquigarrow A\} \preccurlyeq \{? \rightsquigarrow A\}$ and $\{B \rightsquigarrow A, B \rightsquigarrow C\} \preccurlyeq \{B \rightsquigarrow A\}$. Each well-formed network constraint $\mathcal{C}$ represents the set of network topologies that satisfy the (dis)connectivity pairs in $\mathcal{C}$, i.e., $\Gamma(\mathcal{C}) = \{\gamma \mid \mathcal{C}_\Gamma(\gamma) \preccurlyeq \mathcal{C}\}$ where $\mathcal{C}_\Gamma(\gamma) = \{\ell \rightsquigarrow \ell' \mid \ell' \in \gamma(\ell)\} \cup \{\ell \not\rightsquigarrow \ell' \mid \ell' \notin \gamma(\ell)\}$ extracts all one-hop (dis)connectivity information from γ. So the empty network constraint $\{\}$ still denotes all possible topologies over Loc. The negation $\neg\,\mathcal{C}$ of network constraint $\mathcal{C}$ is obtained by negating all its (dis)connectivity pairs. Clearly, if $\mathcal{C}$ is well-formed then so is $\neg\,\mathcal{C}$.

*Constrained labeled transition system*s (CLTSs) provide a semantic model for the operational behavior of MANETs. Let Msg denote a set of messages communicated over a network and ranged over by $\mathfrak{m}$. Let Act be the network send and receive actions with signatures $nsnd : Msg \times Loc$ and $nrcv : Msg$, respectively. The send action $nsnd(\mathfrak{m}, \ell)$ denotes that the message $\mathfrak{m}$ is transmitted from a node with the address ℓ, while the receive action $nrcv(\mathfrak{m})$ denotes that the message $\mathfrak{m}$ is ready to be received. Let $Act_\tau = Act \cup \{\tau\}$, ranged over by η.

Definition 2.1. A *CLTS* is a tuple $\langle S, \Lambda, \rightarrow, s_0 \rangle$, with S a set of states, $\Lambda \subseteq \mathbb{C}^v(Loc) \times Act_\tau$, $\rightarrow \subseteq S \times \Lambda \times S$ a transition relation, and $s_0 \in S$ the initial state. A transition $(s, (\mathcal{C}, \eta), s') \in \rightarrow$ is denoted by $s \xrightarrow{(\mathcal{C},\eta)} s'$.

Generally speaking, the transition $s \xrightarrow{(\mathcal{C},\eta)} s'$ expresses that a MANET protocol in state s with an underlying topology $\gamma \in \Gamma(\mathcal{C})$ can perform action η to evolve to state s'.

The semantics of broadcast communication is defined to be reliable if and if only the nodes that are connected to the sender, as defined by its corresponding network constraint, receive the message. We remark that the status of the links from the receivers to the sender or between two arbitrary receivers are not of importance and hence, they are abstracted away. Therefore, by constructing such network constraints through the semantic rules, reliable communication is brought into our framework.

3. Syntax of *RRBPT*

Let $\mathcal{A}$ denote a countably infinite set of process names which are used as recursion variables in recursive specifications. Besides *network send* and *receive* actions, i.e., $nsnd(\mathfrak{m}, \ell)$ and $nrcv(\mathfrak{m})$, we assume *protocol send* and *receive* actions, denoted by $snd, rcv : Msg$, i.e., parametrized by messages. Furthermore, let $IAct$ be a set of internal actions. ***RRBPT*** is a two-sorted algebra, consisting of protocols and networks. The former defines the set of processes that can be deployed on a node of the network, while the latter defines a MANET network in terms of one node or the parallel composition of more nodes. The limitations on the application of operators are imposed by the sorts. Most of the limitations will be dropped later; the two sorts will be merged into a one-sorted algebra in Section 6. The syntax of ***RRBPT*** is given by the following grammar:

$$\begin{aligned} p ::= &\quad 0 \mid \alpha.p \mid p + p \mid sense(\ell, p, p) \mid \mathfrak{A},\ \mathfrak{A} \stackrel{def}{=} p \\ t ::= &\quad [\![p]\!]_\ell \mid t \parallel t \mid (\nu\ell)t \mid \tau_{\mathfrak{m}}(t) \mid \partial_{\mathfrak{m}}(t) \end{aligned}$$

Deadlock is modeled by 0. The process $\alpha.p$ performs action α and then behaves as p, where α is either an internal action or a protocol send/receive action $snd(\mathfrak{m})/rcv(\mathfrak{m})$. Internal actions are useful in modeling the interactions of a process with other applications running on the same node. Protocol send/receive actions specify the interaction of a process with its data-link layer protocols: these protocols are responsible for transferring messages reliably throughout the network. These actions are turned into their corresponding network ones via the semantics (see Section 4). The process $p_1 + p_1$ behaves non-deterministically as p_1 or p_2. A process name is specified by a recursive equation $\mathfrak{A} \stackrel{def}{=} p$ where $\mathfrak{A} \in \mathcal{A}$ is a name.

The simplest form of a MANET is a node, represented by the network deployment operator $[\![p]\!]_\ell$, denoting process p deployed on a node with the known network address $\ell \neq ?$ (where $?$ denotes the unknown address). A MANET can be composed by putting MANETs in parallel using $\parallel$; the nodes communicate with each other by reliable restricted broadcast.

MANET protocols may behave based on the (non-)existence of a link. A neighbor discovery service can be implemented at the data link layer, by periodically sending *hello* messages and acknowledging such messages received from a neighbor. The sensing operator $sense(\ell', p_1, p_2)$ examines the

status of the link from the node, say with address ℓ, that the sensing is executed on, to the node with the address ℓ'; in case of its existence it behaves as p_1, and otherwise as p_2. For instance, the term $[\![sense(\ell', p_1, p_2)]\!]_\ell$ examines the existence of the link $\ell \rightsquigarrow \ell'$, and then behaves accordingly. As a running example, $P \stackrel{def}{=} sense(B, snd(data_B).P, 0)$ denotes a process that recursively broadcasts a data message $data_B$ as long as it is connected to B; and $Q \stackrel{def}{=} rcv(data_B).deliver.Q$ a process that recursively receives a data message $data$ and then the internal action $deliver$ upon successful receipt of data. The network process $[\![P]\!]_A \parallel [\![Q]\!]_B$ specifies an ad hoc network composed of two nodes with the network addresses A and B deploying processes P and Q, respectively.

The hide operator $(\nu\ell)t$ conceals the address ℓ in the process t, by renaming this address to ? in network send/receive actions. For each message $\mathfrak{m} \in Msg$, the abstraction operator $\tau_{\mathfrak{m}}(t)$ renames network send/receive actions over the message $\mathfrak{m}$ to τ, and the encapsulation operator $\partial_{\mathfrak{m}}(t)$ forbids receiving the message $\mathfrak{m}$. Let $\tau_{\{\mathfrak{m}_1,\ldots,\mathfrak{m}_n\}}(t)$ and $\partial_{\{\mathfrak{m}_1,\ldots,\mathfrak{m}_n\}}(t)$ denote $\tau_{\mathfrak{m}_1}(\ldots(\tau_{\mathfrak{m}_n}(t))\ldots)$ and $\partial_{\mathfrak{m}_1}(\ldots(\partial_{\mathfrak{m}_n}(t))\ldots)$, respectively.

For example, $\tau_{Msg}(\partial_{Msg}([\![P]\!]_A \parallel [\![Q]\!]_B))$ specifies an isolated MANET that cannot receive any message from the environment, while its communications (i.e., send actions) are abstracted away.

4. Semantics of *RRBPT*

Let $PAct$ and $NAct$ denote the set of protocol and network send and receive actions respectively, and $IAct$ the set of internal actions. We assume that $\alpha \in PAct \cup IAct$, $\eta \in NAct \cup IAct \cup \{\tau\}$, $i \in IAct$, and $\iota \in IAct \cup \{\tau\}$. The operational rules in Table 1 induce a CLTS with transitions of the form $t \xrightarrow{\beta} t'$, where $\beta \in \mathbb{C}^v(Loc) \times Act_\tau$ where $Act = NAct \cup IAct$. In these rules, $p \xrightarrow{(\mathcal{C},\ rcv(\mathfrak{m}))}\!\!\!\!\!\!/\ \ $ denotes that there exists no p' such that $p \xrightarrow{(\mathcal{C}',\ rcv(\mathfrak{m}))} p'$ and $\mathcal{C}' \preccurlyeq \mathcal{C}$. The symmetric counterparts of the rules $Choice$, Bro, and Par hold.

Rule $Prefix$ assigns an empty network constraint to each prefixed action, which may be accumulated by further constraints through application of rules $Rcv_{1,2}$ or $Sen_{1,2}$. Rule $Choice$ defines non-deterministic behavior for the protocols. The behavior of a process name is defined in terms of the process in its right-hand definition by the rule Inv.

The rules $Sen_{1,2}$ explain the behavior of the sense operator. In case there is a link to the node with the address ℓ from the node that is running the sense operator, and currently its address is unknown, then it behaves like p_1; in case this link is not present, it behaves like p_2. Therefore, the link status is combined with the network constraint $\mathcal{C}$ generated by its first or second term argument, as given by $Sen_{1,2}$ respectively. For instance, by $Prefix$ and Sen_1, P only generates a $(\{? \rightsquigarrow B\},\ snd(data_B))$-transition.

The rule Int indicates that a node progresses when the deployed protocol on the node performs an internal action. Interaction between the process p and its data-link layer is specified by the rules Snd and Rcv_{1-3}: when p broadcasts a message, it is delivered to the nodes in its transmission range, regardless of their readiness. Rcv_1 specifies that a node $[\![p]\!]_\ell$ with an enabled receive action can perform it successfully if it has a link to a sender (not currently known). In contrast, an enabled receive action cannot be performed if the node is disconnected from the sender (not currently known). However, to make the node *input-enabled* and consequently *non-blocking*, the node still performs its

Table 1. Semantics of *RRBPT* operators.

$$\frac{p_1 \xrightarrow{(\mathcal{C},\alpha)} p_1'}{sense(\ell,p_1,p_2) \xrightarrow{(\{?\rightsquigarrow \ell\}\cup\mathcal{C},\alpha)} p_1'} : Sen_1 \qquad \frac{}{\alpha.p \xrightarrow{(\{\},\alpha)} p} : Prefix$$

$$\frac{p_2 \xrightarrow{(\mathcal{C},\alpha)} p_2'}{sense(\ell,p_1,p_2) \xrightarrow{(\{?\not\rightsquigarrow \ell\}\cup\mathcal{C},\alpha)} p_2'} : Sen_2 \qquad \frac{p_1 \xrightarrow{(\mathcal{C},\alpha)} p_1'}{p_1+p_2 \xrightarrow{(\mathcal{C},\alpha)} p_1'} : Choice$$

$$\frac{p \xrightarrow{(\mathcal{C},\alpha)} p'}{\mathfrak{A} \xrightarrow{(\mathcal{C},\alpha)} p'} : \ Inv,\ \mathfrak{A} \stackrel{def}{=} p \qquad \frac{p \xrightarrow{(\mathcal{C},\alpha)} p'}{p \xrightarrow{(\mathcal{C}',\alpha)} p'} : \ Exe_1,\ \mathcal{C}' \preccurlyeq \mathcal{C}$$

$$\frac{p \xrightarrow{(\mathcal{C},\ snd(\mathfrak{m}))} p'}{[\![p]\!]_\ell \xrightarrow{(\mathcal{C}[\ell/?],\ nsnd(\mathfrak{m},\ell))} [\![p']\!]_\ell} : Snd \qquad \frac{p \xrightarrow{(\mathcal{C},\ rcv(\mathfrak{m}))} p'}{[\![p]\!]_\ell \xrightarrow{(\mathcal{C}[\ell/?]\cup\{?\rightsquigarrow\ell\},\ nrcv(\mathfrak{m}))} [\![p']\!]_\ell} : Rcv_1$$

$$\frac{p \not\xrightarrow{(\mathcal{C},\ rcv(\mathfrak{m}))}}{[\![p]\!]_\ell \xrightarrow{(\mathcal{C}[\ell/?],\ nrcv(\mathfrak{m}))} [\![p]\!]_\ell} : Rcv_3 \qquad \frac{p \xrightarrow{(\mathcal{C},\ rcv(\mathfrak{m}))} p'}{[\![p]\!]_\ell \xrightarrow{(\mathcal{C}[\ell/?]\cup\{?\not\rightsquigarrow\ell\},\ nrcv(\mathfrak{m}))} [\![p]\!]_\ell} : Rcv_2$$

$$\frac{p \xrightarrow{(\mathcal{C},\ i)} p'}{[\![p]\!]_\ell \xrightarrow{(\mathcal{C},\ i)} [\![p']\!]_\ell} : Int \qquad \frac{t_1 \xrightarrow{(\mathcal{C}_1,\ nsnd(\mathfrak{m},\ell))} t_1' \quad t_2 \xrightarrow{(\mathcal{C}_2,\ nrcv(\mathfrak{m}))} t_2'}{t_1 \parallel t_2 \xrightarrow{(\mathcal{C}_1\cup\mathcal{C}_2[\ell/?],\ nsnd(\mathfrak{m},\ell))} t_1' \parallel t_2'} : Bro$$

$$\frac{t \xrightarrow{(\mathcal{C},\eta)} t'}{t \xrightarrow{(\mathcal{C}',\eta)} t'} : \ Exe_2,\ \mathcal{C}' \preccurlyeq \mathcal{C} \qquad \frac{t_1 \xrightarrow{(\mathcal{C}_1,nrcv(\mathfrak{m}))} t_1' \quad t_2 \xrightarrow{(\mathcal{C}_2,nrcv(\mathfrak{m}))} t_2'}{t_1 \parallel t_2 \xrightarrow{(\mathcal{C}_1\cup\mathcal{C}_2,nrcv(\mathfrak{m}))} t_1' \parallel t_2'} : Recv$$

$$\frac{t \xrightarrow{(\mathcal{C},\eta)} t'}{(\nu\ell)t \xrightarrow{(hide(\mathcal{C},\ell),\eta[?/\ell])} (\nu\ell)t'} : Hid \qquad \frac{t_1 \xrightarrow{(\mathcal{C},\iota)} t_1'}{t_1 \parallel t_2 \xrightarrow{(\mathcal{C},\iota)} t_1' \parallel t_2} : Par$$

$$\frac{t \xrightarrow{(\mathcal{C},\eta)} t' \quad \eta \neq nrcv(\mathfrak{m})}{\partial_{\mathfrak{m}}(t) \xrightarrow{(\mathcal{C},\eta)} \partial_{\mathfrak{m}}(t')} : Encap \qquad \frac{t \xrightarrow{(\mathcal{C},\eta)} t'}{\tau_{\mathfrak{m}}(t) \xrightarrow{(\mathcal{C},\tau_{\mathfrak{m}}(\eta))} \tau_{\mathfrak{m}}(t')} : Abs$$

receive action but its state is unchanged, as explained in Rcv_2. If a protocol does not have any enabled receive action $rcv(\mathfrak{m})$ for the network constraint $\mathcal{C}$, then receiving the message has no effect on the node behavior, as explained by Rcv_3. Consequently, this rule makes nodes *input-enabled*, meaning that a node not ready to receive a message will drop it. Therefore, $[\![P]\!]_A$ has a $(\{\}, nrcv(data_B))$-transition by application of this rule.

In rules Snd and Rcv_1, the network constraint $\mathcal{C}$ may have the unknown address due to sensing operators, which is replaced by the address of the deployment operator, i.e., $\mathcal{C}[\ell/?]$. Therefore, by applying Snd to the only transition of P, $[\![P]\!]_A$ generates a $(\{A \rightsquigarrow B\}, nsnd(data_B, A))$-transition.

Exe_1 explains that a behavior that is possible for a network constraint, is also possible for a more restrictive network constraint. This rule is essential for the rule Rcv_3. Without such a rule, Rcv_3 would derive a self-loop with the label $(\{B \not\rightsquigarrow A\}, rcv(data_B))$ for $[\![Q]\!]_B$ as $Q \xrightarrow{(\{B\not\rightsquigarrow A\}, rcv(data_B))}$. While $[\![Q]\!]_B$ also has a $(\{? \rightsquigarrow B\}, nrcv(data_B))$-transition leading to the behavior $[\![y]\!]_B$, where $y \equiv deliver.Q$, by the application of Rcv_1. For topologies that are in common between the two network constraints, $\{? \rightsquigarrow B\}$ and $\{B \not\rightsquigarrow A\}$, $[\![Q]\!]_B$ have a non-deterministic behavior on receiving $data_B$. By Exe_1, Q induces a $(\{B \not\rightsquigarrow A, ? \rightsquigarrow B\}, rcv(data_B))$-transition leading to the behavior y. Finally, $\{B \not\rightsquigarrow A, ? \rightsquigarrow B\} \preceq \{B \not\rightsquigarrow A\}$ makes that the premise $Q \xrightarrow{(\{B\not\rightsquigarrow A\}, rcv(data_B))}$ does not hold, and hence the behavior of $[\![Q]\!]_B$ is deterministic on receive actions. The counterpart of this rule holds for the network processes as defined by the rule Exe_2.

Rule $Recv$ synchronizes the receive actions of processes t_1 and t_2 on message $\mathfrak{m}$, while combining together their (dis)connectivity information in network constraints $\mathcal{C}_1$ and $\mathcal{C}_2$. Rule Bro specifies how a communication occurs between a receiving and a sending process. This rule combines the network constraints, while the unknown location (in the network constraint of the receiving process) is replaced by the concrete address of the sender. In Bro and $Recv$ it is required that the union of network constraints on the transition in the conclusion be well-formed.

The rule Par prevents evolution of sub-networks on network actions, in contrast to lossy settings, and enforces all nodes to specify their localities with respect to the sender before evolving the whole network via $Recv$ or Bro rules. It only allows a process to evolve by performing an internal or a silent action.

For instance, the MANET $[\![P]\!]_A \parallel [\![Q]\!]_B$ can generate the $(\{B \rightsquigarrow A\},\ nsnd(data_B, A))$ transition induced by the deduction tree below, where $y \equiv deliver.Q$:

$$\dfrac{\dfrac{\dfrac{}{P \xrightarrow{(\{\},\ snd(data_B))} P}:Prefix}{P \xrightarrow{(\{?\rightsquigarrow B\},\ snd(data_B))} P}:Sen_1}{[\![P]\!]_A \xrightarrow{(\{A\rightsquigarrow B\},\ nsnd(data_B,A))} [\![P]\!]_A}:Snd \qquad \dfrac{\dfrac{}{Q \xrightarrow{(\{\},\ rcv(data_B))} y}:Prefix}{[\![Q]\!]_B \xrightarrow{(\{?\rightsquigarrow B\},\ nrcv(data_B))} [\![y]\!]_B}:Rcv_1$$

$$[\![P]\!]_A \parallel [\![Q]\!]_B \xrightarrow{(\{B\rightsquigarrow A\},\ nsnd(data_B,A))} [\![P]\!]_A \parallel [\![y]\!]_B \quad :Bro$$

Rule Hid replaces every occurrence of ℓ in the network constraint and action of β by ?, and hence hides activities of a node with address ℓ from external observers. As the receipt by unknown addresses does not provide any information useful for reasoning about the behavior of MANETs, $hide(\mathcal{C}, \ell)$ also removes (dis)connectivity pairs with unknown receivers while replacing ℓ by ?:

$$hide(\mathcal{C}, \ell) = \{\ell_1 \rightsquigarrow \ell_2 \in \mathcal{C}[?/\ell] \mid \ell_2 \neq ?\} \cup \{\ell_1 \not\rightsquigarrow \ell_2 \in \mathcal{C}[?/\ell] \mid \ell_2 \neq ?\}.$$

Therefore the address ? only represents a sending node with an unknown address. According to Abs, the abstraction operator $\tau_{\mathfrak{m}}$ converts all network send and receive actions over the message $\mathfrak{m}$ to τ and leaves other actions unaffected, as defined by the function $\tau_{\mathfrak{m}}(\eta)$. The encapsulation operator $\partial_{\mathfrak{m}}$ disallows receiving the message $\mathfrak{m}$, as specified by $Encap$.

The semantics of *RRBPT* was first introduced in [18] with the aim of defining CLTSs with negative connectivity pairs to illustrate their benefit for model checking MANET protocols. In this research, we modify its semantics to properly define the behavior of MANETs in the reliable setting. To this end, the rules of the sensing operator have been modified substantially. The semantics of the sensing operator in [18] makes $[\![P]\!]_A$ move by $(\{B \not\rightsquigarrow A, ? \rightsquigarrow A\}, nrcv(data_B))$ and $(\{B \not\rightsquigarrow A, ? \not\rightsquigarrow A\}, nrcv(data_B))$ to $[\![0]\!]_A$ while here it has a self-loop with the label of $(\{B \not\rightsquigarrow A\}, nrcv(data_B))$. In other words, the chance of sending $data_B$ is lost after dropping a received message of $data_B$. Such a drawback is resolved by the modified rule Rcv_3 and removing two previous rules of the sensing operator. Furthermore, the third rule of receive actions has been modified by adding the conjunct $\mathcal{C}' \preceq \mathcal{C}$ as a premise to avoid dropping a message when it can be processed.

5. Rooted branching reliable computed network bisimilarity

Terms of the lossy framework *RBPT* are considered modulo rooted branching computed network bisimilarity [21]. This equivalence relation is defined using the following notations:

- $\Rightarrow$ denotes the reflexive and transitive closure of unobservable actions:
  - $t \Rightarrow t$;
  - if $t \xrightarrow{(\mathcal{C},\tau)} t'$ for some arbitrary network constraint $\mathcal{C}$ and $t' \Rightarrow t''$, then $t \Rightarrow t''$.
- $t \xrightarrow{\langle(\mathcal{C},\eta)\rangle} t'$ iff $t \xrightarrow{(\mathcal{C},\eta)} t'$ or $t \xrightarrow{(\mathcal{C}[\ell/?],\eta[\ell/?])} t'$ and η is of the form $nsnd(\mathfrak{m}, ?)$ for some $\mathfrak{m}$.

Intuitively $t \Rightarrow t'$ expresses that after a number of communications, t can behave like t'. Furthermore, as the activities of a node with an unknown address can be matched to those with a known address in our equivalence relation, we define the notion of *counterpart* action, denoted by $\langle(\mathcal{C}, \eta)\rangle$. For example, an action like $(\{? \rightsquigarrow B\}, nsnd(req, ?))$ can be matched to an action like $(\{A \rightsquigarrow B\}, nsnd(req, A))$, which can be considered as its counterpart denoted by $\langle(\{? \rightsquigarrow B\}, nsnd(req, ?))\rangle$.

Definition 5.1. A binary relation $\mathcal{R}$ on *RBPT* terms is a branching computed network simulation if $t_1 \mathcal{R} t_2$ and $t_1 \xrightarrow{(\mathcal{C},\eta)} t_1'$ implies that either:

- η is of the form $nrcv(\mathfrak{m})$ or τ, and $t_1' \mathcal{R} t_2$; or
- there are t_2' and t_2'' such that $t_2 \Rightarrow t_2'' \xrightarrow{\langle(\mathcal{C},\eta)\rangle} t_2'$, where $t_1 \mathcal{R} t_2''$ and $t_1' \mathcal{R} t_2'$.

$\mathcal{R}$ is a branching computed network bisimulation if $\mathcal{R}$ and $\mathcal{R}^{-1}$ are branching computed network simulations. Two terms t_1 and t_2 are branching computed network bisimilar, denoted by $t_1 \simeq_b t_2$, if $t_1 \mathcal{R} t_2$ for some branching computed network bisimulation relation $\mathcal{R}$.

This definition distinguishes process terms according to their abilities to broadcast messages, and therefore, MANET protocols that can only receive are treated as deadlock as they cannot send any observable message.

Definition 5.2. Terms t_1 and t are *rooted branching computed network bisimilar*, written $t_1 \simeq_{rb} t_2$, if:

- $t_1 \xrightarrow{(\mathcal{C},\eta)} t_1'$ implies there is a t_2' such that $t_2 \xrightarrow{\langle(\mathcal{C},\eta)\rangle} t_2'$ and $t_1' \simeq_b t_2'$;
- $t_2 \xrightarrow{(\mathcal{C},\eta)} t_2'$ implies there is a t_1' such that $t_1 \xrightarrow{\langle(\mathcal{C},\eta)\rangle} t_1'$ and $t_1' \simeq_b t_2'$.

Rooted branching computed network bisimilarity does not constitute a congruence with respect to the *RRBPT* operators. We still want that a receiving MANET (after its first action) be equivalent to deadlock. In this setting, still $[\![0]\!]_A \simeq_b [\![rcv(\mathfrak{m}).0]\!]_A$, but $[\![0]\!]_A \parallel [\![snd(\mathfrak{m}).0]\!]_B \not\simeq_b [\![rcv(\mathfrak{m}).0]\!]_A \parallel [\![snd(\mathfrak{m}).0]\!]_B$, since by application of $Rcv_{1,2}$, Snd, and Bro:

$$[\![rcv(\mathfrak{m}).0]\!]_A \parallel [\![snd(\mathfrak{m}).0]\!]_B \xrightarrow{(\{B \not\rightsquigarrow A\}, nsnd(\mathfrak{m},B))} [\![rcv(\mathfrak{m}).0]\!]_A \parallel [\![0]\!]_B$$
$$[\![rcv(\mathfrak{m}).0]\!]_A \parallel [\![snd(\mathfrak{m}).0]\!]_B \xrightarrow{(\{B \rightsquigarrow A\}, nsnd(\mathfrak{m},B))} [\![0]\!]_A \parallel [\![0]\!]_B$$

while by application of Rcv_3, Snd, Bro:

$$[\![0]\!]_A \parallel [\![snd(\mathfrak{m}).0]\!]_B \xrightarrow{(\{\}, nsnd(\mathfrak{m},B))} [\![0]\!]_A \parallel [\![0]\!]_B$$

which cannot be matched to any transition of $[\![rcv(\mathfrak{m}).0]\!]_A \parallel [\![snd(\mathfrak{m}).0]\!]_B$ according to the second condition of Definition 5.1. However, we observe that the $(\{\}, nsnd(\mathfrak{m}, B))$-transition can be matched to the transition sets of actions $(\{B \not\rightsquigarrow A\}, nsnd(\mathfrak{m}, B))$ and $(\{B \rightsquigarrow A\}, nsnd(\mathfrak{m}, B))$, as the network constraints $\{B \not\rightsquigarrow A\}$ and $\{B \rightsquigarrow A\}$ provide a partitioning of $\{\}$ while the resulting states of their corresponding transitions are equivalent. Thus, we revise our Definition 5.1 by generalizing its second condition.

Intuitively, two MANETs are equivalent if they have the same observable behaviors for all possible underlying topologies. In the lossy setting, the observable behaviors exclude receive actions, as the node $[\![rcv(a).snd(a).0]\!]_A$ can be distinguished from $[\![rcv(a).0]\!]_A$ due to its capability to send a after its receipt. However, the capability of receiving messages implicitly defines a restriction on the underlying topology. For instance, the sending action $snd(a)$ in $[\![rcv(a).snd(a).0]\!]_A$ is only possible if the node in question was previously connected to a sender and successfully received a. Thus to distinguish $[\![rcv(a).snd(a).0]\!]_A$ from $[\![snd(a).0]\!]_A$, receive actions are included in the observables in the reliable setting. Furthermore, as dropping a message may have the same effect as its processing (as explained above), a transition cannot be matched in the same way as in Definition 5.1 and it may be matched to multiple transitions. A partitioning of a network constraint $\mathcal{C}$ consists of network constraints $\mathcal{C}_1, \ldots, \mathcal{C}_n$ such that $\forall i, j \leq n\,(i \neq j \Rightarrow \Gamma(\mathcal{C}_i) \cap \Gamma(\mathcal{C}_j) = \emptyset) \wedge \bigcup_{k=1}^{n} \Gamma(\mathcal{C}_k) = \Gamma(\mathcal{C})$.

Definition 5.3. A binary relation $\mathcal{R}$ on *RRBPT* terms is a branching reliable computed network simulation if $t_1 \mathcal{R} t_2$ and $t_1 \xrightarrow{(\mathcal{C},\eta)} t_1'$ imply that either:

- η is a τ action, and $t_1' \mathcal{R} t_2$; or
- there are $s_1'', \ldots, s_k''$ and $s_1', \ldots, s_k'$ for some $k > 0$ such that $\forall i \leq k (t_2 \Rightarrow s_i'' \xrightarrow{\langle(\mathcal{C}_i,\eta)\rangle} s_i'$, with $t_1 \mathcal{R} s_i''$ and $t_1' \mathcal{R} s_i'$), and $\langle\mathcal{C}_1\rangle, \ldots, \langle\mathcal{C}_k\rangle$ constitute a partitioning of $\langle\mathcal{C}\rangle$.

$\mathcal{R}$ is a branching reliable computed network bisimulation if $\mathcal{R}$ and $\mathcal{R}^{-1}$ are branching reliable computed network simulations. Two terms t_1 and t_2 are branching reliable computed network bisimilar, denoted by $t_1 \simeq_{br} t_2$, if $t_1\ \mathcal{R}\ t_2$ for some branching reliable computed network bisimulation $\mathcal{R}$.

Trivially $(t_1 \simeq_b t_2) \Rightarrow (t_1 \simeq_{br} t_2)$.

Theorem 5.4. Branching reliable computed network bisimilarity is an equivalence.

See Section A for the proof of this theorem.

Definition 5.5. Two terms t_1 and t are *rooted branching reliable computed network bisimilar*, written $t_1 \simeq_{rbr} t_2$, if:

- $t_1 \xrightarrow{(\mathcal{C},\eta)} t_1'$ implies there is a t_2' such that $t_2 \xrightarrow{\langle(\mathcal{C},\eta)\rangle} t_2'$ and $t_1' \simeq_{br} t_2'$;
- $t_2 \xrightarrow{(\mathcal{C},\eta)} t_2'$ implies there is a t_1' such that $t_1 \xrightarrow{\langle(\mathcal{C},\eta)\rangle} t_1'$ and $t_1' \simeq_{br} t_2'$.

Corollary 5.6. Rooted branching reliable computed network bisimilarity is an equivalence.

Corollary 5.6 is an immediate consequence of Theorem 5.4 and Definition 5.5.

Theorem 5.7. Rooted branching reliable computed network bisimilarity is a congruence for *RRBPT* operators.

See Section B for the proof.

6. Axiomatization for *RRBPT*

To provide a sound and complete axiomatization for closed *RRBPT* terms with respect to rooted branching reliable computed network bisimilarity, the framework should be extended with the computed network terms, i.e., $(\mathcal{C},\eta).t$ which expresses that the action η is possible for the topologies belonging to $\mathcal{C}$, in the same way as [21]. This prefix operator is helpful to transform protocol send/receive actions into their corresponding network ones. Furthermore, it borrows the operators *left merge* ($\between$) and *communication merge* from the process algebra *ACP* [4] to axiomatize parallel composition. Note that the interleaving semantics for parallel composition is only valid for internal and unobservable actions (see rule Par). To axiomatize the behavior of nodes while being input-enabled, we also exploit two novel auxiliary operators.

RRBPT is extended with new operators and called *Reliable Computed Network Process Theory* (*RCNT*), while its two-sorted limitation is dropped. Its syntax contains:

$$\begin{aligned} t ::=\ & 0 \mid \beta.t \mid t+t \mid \mathfrak{A}\,,\mathfrak{A} \stackrel{def}{=} t \mid t \mid t \mid t \between t \mid t \parallel t \mid rec\mathfrak{A}\cdot t \\ & sense(\ell,t,t) \mid (\nu\ell)t \mid \tau_{\mathfrak{m}}(t) \mid \partial_{\mathfrak{m}}(t) \mid \ell : t : t \mid \mathcal{C} \rhd t \mid [\![t]\!]_{\ell} \end{aligned}$$

The prefix operator in $\beta.t$ again denotes a process which performs β and then behaves as t. The action β can now be of two types: either an internal action or a send/receive action $snd(\mathfrak{m})/rcv(\mathfrak{m})$,

denoted by α, or actions of the form $(\mathcal{C}, nrcv(\mathfrak{m}))$, $(\mathcal{C}, nsnd(\mathfrak{m}, \ell))$ and $(\mathcal{C}, \tau)$, denoted by $(\mathcal{C}, \eta)$, where the first two actions are called the network receive and send actions, respectively. The new operator $\ell : t_1 : t_2$, so-called *local deployment*, defines the behavior of process t_2 deployed at the network address ℓ while it only defines the input-enabledness feature with regard to t_1. In cases that it should drop a message (i.e., processing the message has not been defined by t_2), it behaves as t_1. This operator is helpful to axiomatize the behavior of the deployment operator in the reliable setting. To axiomatize the behavior of the $sense$ operator, the framework is extended with the *topology restriction* operator $\mathcal{C} \rhd t$ which restricts the behavior of t by taking restrictions of $\mathcal{C}$ into account.

Due to the input-enabledness feature of nodes, their behavior is recursive: upon receiving a message for which no receive action has been defined, a node drops the message. To this aim, we exploit the recursion operator $rec\mathfrak{A} \cdot t$, which specifies the *solution* of the process name $\mathfrak{A}$, defined by the equation $\mathfrak{A} \stackrel{def}{=} t$. The process term $t_{\mathfrak{A}}$ is a solution of the equation $\mathfrak{A} \stackrel{def}{=} t$ if the replacement of $\mathfrak{A}$ by $t_{\mathfrak{A}}$ on both sides of the equation results in equal terms, i.e. $t_{\mathfrak{A}} \simeq_{rb} t[t_{\mathfrak{A}}/\mathfrak{A}]$. As we are interested in equations with exactly one solution, we define a guardedness criterion for network names, in the same way as [21]. A free occurrence of a network name A in t is called *guarded* if this occurrence is in the scope of an action prefix operator (not $(\mathcal{C}, \tau)$ prefix) and not in the scope of an abstraction operator [1]; in other words, there is a subterm $(\mathcal{C}, \eta).t'$ in t such that $\eta \neq \tau$, and $\mathfrak{A}$ occurs in t'. $\mathfrak{A}$ is *(un)guarded* in t if (not) every free occurrence of $\mathfrak{A}$ in t is guarded. A *RCNT* term t is *guarded* if for every subterm $rec\mathfrak{A} \cdot t'$, $\mathfrak{A}$ is guarded in t'. This guardedness criterion ensures that any guarded recursive term has a unique solution.

A term is grammatically well-defined if its processes deployed at a network address through either a network or local deployment operator, are only defined by action prefix, choice, sense, and process names.

The operational semantic rules of the new operators are given in Table 2 while the application of the rules $Prefix$ and $Choice$ in Table 1 is extended to η-like actions. In these rules, $t \xrightarrow{\not\ nrcv(\mathfrak{m})}$ denotes that there exists no t' such that $t \xrightarrow{(\mathcal{C}',\ nrcv(\mathfrak{m}))} t'$ for some network constraint $\mathcal{C}'$. The behavior of the local deployment operator is almost similar to the deployment operator. Its rules $Inter'_1$ and $Inter'_2$ are almost the same as Snd and Rcv_1, respectively. However, it substitutes $Inter'_3$ for Rcv_2 by which it only adds transitions containing the disconnectivity pair $? \not\rightarrow \ell$ for those possible receive actions of t_2 (generated by Rcv_1). Rules $Choice'$, Inv', and $Sen'_{1,2}$ are also the same as $Choice$, Inv, and $Sen_{1,2}$, receptively. The difference between the rules of the deployment and local deployment is that the behavior of the deployment operator is derived regarding the behavior of the deployed protocol by the rules Snd and Rcv_{1-3} while the behavior of the local deployment is defined in terms of the structure of the deployed protocol. Rules $Sen'_{3,4}$ make the behavior of $sense(\ell', t_1, t_2)$ input-enabled toward receive actions that are possible by t_1 but not t_2 and vice versa. The constraints of the topology restriction operator $\mathcal{C} \rhd t$ is added to the behaviors of t as explained by the rule TR.

The main differences of extended *RCNT* with *CNT* are that its deployed nodes are input-enabled and its communication primitive is reliable. We use the notation $\sum_{m \in M} t$ to define $t[m_1/\mathfrak{m}] + \ldots + t[m_k/\mathfrak{m}]$, where $M = \{m_1, \ldots, m_k\}$. Furthermore, $if(b, t_1, t_2)$ behaves as t_1 if the condition b holds and otherwise as t_2.

Table 2. Semantics of the new operators of *RCNT*

$$\frac{}{\ell : t_1 : snd(\mathfrak{m}).t_2 \xrightarrow{(\{\},\ nsnd(\mathfrak{m},\ell))} [\![t_2]\!]_\ell} : Inter'_1 \qquad \frac{t[rec\mathfrak{A} \cdot t/\mathfrak{A}] \xrightarrow{(\mathcal{C},\eta)} t'}{rec\mathfrak{A} \cdot t \xrightarrow{(\mathcal{C},\eta)} t'} : Rec$$

$$\frac{}{\ell : t_1 : rcv(\mathfrak{m}).t_2 \xrightarrow{(\{?\rightsquigarrow\ell\},\ nrcv(\mathfrak{m}))} [\![t_2]\!]_\ell} : Inter'_2$$

$$\frac{}{\ell : t_1 : rcv(\mathfrak{m}).t_2 \xrightarrow{(\{?\not\rightsquigarrow\ell\},\ nrcv(\mathfrak{m}))} t_1} : Inter'_3 \qquad \frac{t \xrightarrow{(\mathcal{C}',\eta)} t'}{\mathcal{C} \rhd t \xrightarrow{(\mathcal{C}' \cup \mathcal{C},\eta)} t'} : TR$$

$$\frac{\ell : t_3 : t_i \xrightarrow{(\mathcal{C},\ \eta)} t'_i \ \ i \in \{1,2\}}{\ell : t_3 : t_1 + t_2 \xrightarrow{(\mathcal{C},\eta)} t'_i} : Choice' \qquad \frac{\ell : t_3 : t_1 \xrightarrow{(\mathcal{C},\eta)} t'_1}{\ell : t_3 : sense(\ell', t_1, t_2) \xrightarrow{(\{\ell \rightsquigarrow \ell'\} \cup \mathcal{C},\eta)} t'_1} : Sen'_1$$

$$\frac{\ell : t_1 : t_2 \xrightarrow{(\mathcal{C},\ \eta)} t'_2}{\ell : t_1 : \mathfrak{A} \xrightarrow{(\mathcal{C},\eta)} t'_2} : Inv',\ \mathfrak{A} \stackrel{def}{=} t_2 \qquad \frac{\ell : t_3 : t_2 \xrightarrow{(\mathcal{C},\eta)} t'_2}{\ell : t_3 : sense(\ell', t_1, t_2) \xrightarrow{(\{\ell \not\rightsquigarrow \ell'\} \cup \mathcal{C},\eta)} t'_2} : Sen'_2$$

$$\frac{\ell : t_3 : t_1 \xrightarrow{nrcv(\mathfrak{m})} \quad \ell : t_3 : t_2 \xrightarrow{(\mathcal{C},\ nrcv(\mathfrak{m}))} t'_2}{\ell : t_3 : sense(\ell', t_1, t_2) \xrightarrow{(\{\ell \rightsquigarrow \ell'\},\ nrcv(\mathfrak{m}))} t_3} : Sen'_3$$

$$\frac{\ell : t_3 : t_1 \xrightarrow{(\mathcal{C},\ nrcv(\mathfrak{m}))} t'_1 \quad \ell : t_3 : t_2 \xrightarrow{nrcv(\mathfrak{m})}}{\ell : t_3 : sense(\ell', t_1, t_2) \xrightarrow{(\{\ell \not\rightsquigarrow \ell'\},\ nrcv(\mathfrak{m}))} t_3} : Sen'_4$$

$$\frac{t_1 \xrightarrow{(\mathcal{C}_1, nrcv(\mathfrak{m}))} t'_1 \quad t_2 \xrightarrow{(\mathcal{C}_2, nrcv(\mathfrak{m}))} t'_2}{t_1 \mid t_2 \xrightarrow{(\mathcal{C}_1 \cup \mathcal{C}_2, nrcv(\mathfrak{m}))} t'_1 \parallel t'_2} : Sync_1 \qquad \frac{t_1 \xrightarrow{(\mathcal{C},\iota)} t'_1}{t_1 \between t_2 \xrightarrow{(\mathcal{C},\iota)} t'_1 \parallel t_2} : LExe$$

$$\frac{t_i \xrightarrow{(\mathcal{C}_1, nsnd(\mathfrak{m},\ell))} t'_i \quad t_{3-i} \xrightarrow{(\mathcal{C}_2, nrcv(\mathfrak{m}))} t'_{3-i} \quad i \in \{1,2\}}{t_1 \mid t_2 \xrightarrow{(\mathcal{C}_1 \cup \mathcal{C}_2[\ell/?], nsnd(\mathfrak{m},\ell))} t'_1 \parallel t'_2} : Sync_2$$

The axioms regarding the choice, deployment, left and communication merge, and parallel operators are given in Table 3. The axioms Ch_{1-4}, Br, $LM_{2,3}$ and S_{1-4} are standard (cf. [35]). The axiom Ch_5 denotes that a network send action whose sender address is unknown can be removed if its counterpart action exists, i.e., a send action with the same message and receivers but with a known sender address (see Section 5). The axiom Ch_6 explains that a more liberal network constraint allows more behavior. Axioms Dep_{0-7}, $LM'_{1,2}$, and $TRes_{1-5}$ are new in comparison with the lossy setting of [21]. The axiom $(\mathcal{C}, \eta).t_1 \between t_2 = (\mathcal{C}, \eta).(t_1 \parallel t_2)$ has been replaced by $LM'_{1,2}$ which only allow internal or unobservable actions of the left operand to be performed.

Table 3. Axioms for the choice, deployment, left and communication merge, and parallel operators. The sets M_1 and M_2 denote $Message(t_2,\emptyset) \setminus Message(t_1,\emptyset)$ and $Message(t_1,\emptyset) \setminus Message(t_2,\emptyset)$ respectively.

Ch_1	$0 + t = t$	Ch_2	$t_1 + t_2 = t_2 + t_1$
Ch_3	$t_1 + (t_2 + t_3) = (t_1 + t_2) + t_3$	Ch_4	$t + t = t$
Ch_5	$(\mathcal{C}, nsnd(\mathfrak{m}, ?)).t + \langle(\mathcal{C}, nsnd(\mathfrak{m}, ?))\rangle.t = \langle(\mathcal{C}, nsnd(\mathfrak{m}, ?))\rangle.t$		
Ch_6	$(\mathcal{C}_1, \eta).t + (\mathcal{C}_2, \eta).t = (\mathcal{C}_1, \eta).t,\ \mathcal{C}_2 \preccurlyeq \mathcal{C}_1$		
Dep_0	$[\![t]\!]_\ell = rec\mathfrak{Q} \cdot \sum_{\mathfrak{m}' \notin Message(t,\emptyset)}(\{\}, nrcv(\mathfrak{m}')).\mathfrak{Q} + \ell : \mathfrak{Q} : t$		
Dep_2	$\ell : t' : rcv(\mathfrak{m}).t = (\{? \not\rightsquigarrow \ell\}, nrcv(\mathfrak{m})).t' + (\{? \rightsquigarrow \ell\}, nrcv(\mathfrak{m})).[\![t]\!]_\ell$		
Dep_7	$\ell : t_3 : sense(\ell', t_1, t_2) = \sum_{\mathfrak{m}' \in M_1}(\{\ell \rightsquigarrow \ell'\}, nrcv(\mathfrak{m}')).t_3$ $+ \sum_{\mathfrak{m}' \in M_2}(\{\ell \not\rightsquigarrow \ell'\}, nrcv(\mathfrak{m}')).t_3 + \{\ell \rightsquigarrow \ell'\} \rhd \ell : t_3 : t_1 + \{\ell \not\rightsquigarrow \ell'\} \rhd \ell : t_3 : t_2$		
Dep_1	$\ell : t' : snd(\mathfrak{m}).t = (\{\}, nsnd(\mathfrak{m}, \ell)).[\![t]\!]_\ell$	Dep_6	$\ell : t' : i.t = (\{\}, i).[\![t]\!]_\ell$
Dep_3	$\ell : t_3 : t_1 + t_2 = \ell : t_3 : t_1 + \ell : t_3 : t_2$	Dep_4	$\ell : t : 0 = 0$
Dep_5	$\ell : t' : \mathfrak{A} = \ell : t' : t,\ \mathfrak{A} \stackrel{def}{=} t$		
$TRes_1$	$\mathcal{C}_1 \rhd (\mathcal{C}_2, \eta).t = (\mathcal{C}_1 \cup \mathcal{C}_2, \eta).t,\ \text{if } \mathcal{C}_1 \cup \mathcal{C}_2 \in \mathbb{C}^v(Loc)$		
$TRes_2$	$\mathcal{C} \rhd (t_1 + t_2) = (\mathcal{C} \rhd t_1) + (\mathcal{C} \rhd t_2)$	$TRes_3$	$\mathcal{C} \rhd rec\mathfrak{A} \cdot t = rec\mathfrak{A} \cdot (\mathcal{C} \rhd t)$
$TRes_4$	$\mathcal{C} \rhd \mathfrak{A} = \mathfrak{A}$	$TRes_5$	$\mathcal{C} \rhd 0 = 0$
Br	$t_1 \parallel t_2 = t_1 \between t_2 + t_2 \between t_1 + t_1 \mid t_2$	S_1	$t_1 \mid t_2 = t_2 \mid t_1$
LM'_1	$(\mathcal{C}, \eta).t_1 \between t_2 = 0,\ \eta \notin IAct \cup \{\tau\}$	S_2	$(t_1 + t_2) \mid t_3 = t_1 \mid t_3 + t_2 \mid t_3$
LM_2	$(t_1 + t_2) \between t_3 = t_1 \between t_3 + t_2 \between t_3$	S_3	$0 \mid t = 0$
LM_3	$0 \between t = 0$	S_4	$(\mathcal{C}, \eta).t_1 \mid t_2 = 0,\ \eta \in IAct \cup \{\tau\}$
LM'_2	$(\mathcal{C}, \eta).t_1 \between t_2 = (\mathcal{C}, \eta).(t_1 \parallel t_2),\ \eta \in IAct \cup \{\tau\}$		
$Sync_1$	$(\mathcal{C}_1, nsnd(\mathfrak{m}_1, \ell)).t_1 \mid (\mathcal{C}_2, nrcv(\mathfrak{m}_2)).t_2 =$ $if((\mathfrak{m}_1 = \mathfrak{m}_2), (\mathcal{C}_1 \cup \mathcal{C}_2[\ell/?], nsnd(\mathfrak{m}_1, \ell)).t_1 \parallel t_2, 0)$		
$Sync_2$	$(\mathcal{C}_1, nrcv(\mathfrak{m}_1)).t_1 \mid (\mathcal{C}_2, nrcv(\mathfrak{m}_2)).t_2 = if((\mathfrak{m}_1 = \mathfrak{m}_2), (\mathcal{C}_1 \cup \mathcal{C}_2, nrcv(\mathfrak{m}_1)).t_1 \parallel t_2, 0)$		
$Sync_3$	$(\mathcal{C}_1, nsnd(\mathfrak{m}_1, \ell_1)).t_1 \mid (\mathcal{C}_2, nsnd(\mathfrak{m}_2, \ell_2)).t_2 = 0$		

To axiomatize the behavior of a node considering the input-enabledness feature, we need to find the messages that it cannot currently respond to and then add a summand which receives those message without processing them. To this aim, axiom Dep_0 expresses the behavior of $[\![t]\!]_\ell$ as a recursive specification which drops messages that it does not handle with the help of the auxiliary function $Message(t, \mathcal{S})$, and the behavior of t with the help of the local deployment operator $\ell : \mathfrak{Q} : t$. The function $Message(t, \mathcal{S})$ returns the set of messages that can be currently processed by t and is defined using structural induction:

$$Message(0, \mathcal{S}) = \emptyset$$
$$Message(i.t, \mathcal{S}) = \emptyset,\ i \in IAct$$
$$Message(snd(\mathfrak{m}).t, \mathcal{S}) = \emptyset$$
$$Message(rcv(\mathfrak{m}).t, \mathcal{S}) = \{\mathfrak{m}\}$$
$$Message(t_1 + t_2, \mathcal{S}) = Message(t_1, \mathcal{S}) \cup Message(t_2, \mathcal{S})$$
$$Message(sense(\ell, t_1, t_2), \mathcal{S}) = Message(t_1, \mathcal{S}) \cup Message(t_2, \mathcal{S})$$
$$Message(\mathfrak{A}, \mathcal{S}) = Message(t, \mathcal{S} \cup \{\mathfrak{A}\}),\ \mathfrak{A} \notin \mathcal{S}, \mathfrak{A} \stackrel{def}{=} t$$
$$Message(\mathfrak{A}, \mathcal{S}) = \emptyset,\ \mathfrak{A} \in \mathcal{S}$$

where $\mathcal{S}$ keeps track of process names whose right-hand definitions have been examined. We remark that Dep_0 extends the deployment behavior of the lossy setting with the input enabledness feature with the help of operator $\ell : \mathfrak{Q} : t$. The axioms Dep_{1-7} specify the behavior of the operator $\ell : t_1 : t_2$. Axiom Dep_1 defines the interaction between the network and data link layers. The protocol send action (at the network layer) is transformed into its network version (at the data link layer). Axiom Dep_2 indicates that when ℓ is connected to a sender (which is unknown yet), the receive action is successful and its behavior proceeds as $[\![t]\!]_\ell$. Otherwise, the receive action is unsuccessful and its behavior is defined by t'. Axioms $Dep_{3,4,5}$ express the effect of the local deployment on choice, deadlock, and process names, respectively while axioms $Dep_{6,7}$ define its effect on the prefixed internal actions and $sense$ operator, respectively.

The behavior of the topology restriction operator is defined by the axioms $TRes_{1-5}$ in Table 3. Axiom $TRes_1$ considers the restrictions of $\mathcal{C}_1$ by integrating its restrictions with $\mathcal{C}_2$ in the computed network term $(\mathcal{C}_2, \eta).t$ if $\mathcal{C}_1 \cup \mathcal{C}_2$ is well-formed. Axiom $TRes_2$ defines that topology restriction can be distributed over the choice operator. Axiom $TRes_3$ expresses that the topology restriction operator can be moved inside and outside of a recursion operator. Axioms $TRes_{4,5}$ explain that the topology restriction operator has no effect on a process name and deadlock, respectively.

For instance, the behavior of the MANET $[\![P]\!]_A$, where $P \stackrel{def}{=} sense(B, snd(data_B).P, 0)$, $Msg = \{data_B\}$, is simplified as:

$$[\![P]\!]_A =^{Dep_{0,5}}$$
$$rec\mathfrak{Q} \cdot (\{\}, nrcv(data_B)).\mathfrak{Q} + A : \mathfrak{Q} : sense(B, snd(data_B).P, 0) =^{Dep_7}$$
$$rec\mathfrak{Q} \cdot (\{\}, nrcv(data_B)).\mathfrak{Q} + \{A \rightsquigarrow B\} \rhd A : \mathfrak{Q} : snd(data_B).P + \{A \not\rightsquigarrow B\} \rhd A : \mathfrak{Q} : 0 =^{Dep_{1,4}}$$
$$rec\mathfrak{Q} \cdot (\{\}, nrcv(data_B)).\mathfrak{Q} + \{A \rightsquigarrow B\} \rhd (\{\}, nsnd(data_B, A)).\mathfrak{Q} + \{B \not\rightsquigarrow A\} \rhd 0 =^{TRes_{1,5}}$$
$$rec\mathfrak{Q} \cdot (\{\}, nrcv(data_B)).\mathfrak{Q} + (\{A \rightsquigarrow B\}, nsnd(data_B, A)).\mathfrak{Q}$$

The behavior of $[\![Q]\!]_B$, where $Q \stackrel{def}{=} rcv(data_B).deliver.Q$, is equated to:

$$[\![Q]\!]_B =^{Dep_{0,5}}$$
$$rec\mathfrak{Q} \cdot A : \mathfrak{Q} : rcv(data_B).deliver.Q =^{Dep_2}$$
$$rec\mathfrak{Q} \cdot (\{? \rightsquigarrow B\}, nrcv(data_B)).[\![deliver.Q]\!]_A + (\{? \not\rightsquigarrow B\}, nrcv(data_B)).\mathfrak{Q}$$

Table 4. Axiomatization of hiding, abstraction and encapsulation operators.

Res_1	$(\nu\ell)(t_1+t_2)=(\nu\ell)t_1+(\nu\ell)t_2$	Res_3	$(\nu\ell)0=0$
Res_2	$(\nu\ell)(\mathcal{C},\eta).t=(hide(\mathcal{C},\ell),\eta[?/\ell]).(\nu\ell)t$		
Ecp_1	$\partial_m((\mathcal{C},nsnd(\mathfrak{m},\ell)).t)=(\mathcal{C},nsnd(\mathfrak{m},\ell)).\partial_m(t)$		
Ecp_2	$\partial_m((\mathcal{C},nrcv(\mathfrak{m})).t)=if((m\neq\mathfrak{m}),(\mathcal{C},nrcv(\mathfrak{m})).\partial_m(t),0)$		
Abs_1	$\tau_m((\mathcal{C},nrcv(\mathfrak{m})).t)=if((m=\mathfrak{m}),(\mathcal{C},\tau).\tau_m(t),(\mathcal{C},nrcv(\mathfrak{m})).\tau_m(t))$		
Abs_2	$\tau_m((\mathcal{C},nsnd(\mathfrak{m},\ell)).t)=if((m=\mathfrak{m}),(\mathcal{C},\tau).\tau_m(t),(\mathcal{C},nsnd(\mathfrak{m},\ell)).\tau_m(t))$		
Abs_3	$\tau_m(t_1+t_2)=\tau_m(t_1)+\tau_m(t_2)$	Ecp_3	$\partial_m(t_1+t_2)=\partial_m(t_1)+\partial_m(t_2)$
Abs_4	$\tau_m(0)=0$	Ecp_4	$\partial_m(0)=0$
T_1	$(\mathcal{C}',\eta).((\mathcal{C}_1,\eta).t+(\mathcal{C}_2,\eta).t+t')=(\mathcal{C}',\eta).((\mathcal{C},\eta).t+t')$ iff $\exists\ell,\ell'\in Loc)\cdot(\mathcal{C}_1=\mathcal{C}\cup\{\ell\rightsquigarrow\ell'\}\wedge\mathcal{C}_2=\mathcal{C}\cup\{\ell\not\rightsquigarrow\ell'\}$		
T_2	$(\mathcal{C},\eta).((\mathcal{C}',\tau).(t_1+t_2)+t_2)=(\mathcal{C},\eta).(t_1+t_2)$		

The axioms of hiding and encapsulation are given in Table 4. Axiom T_1 accumulates the network constraints that constitute a partitioning while T_2 removes a τ action which preserves the behavior of a network after some topology changes. The remaining axioms in this table are similar to the lossy setting.

Axioms for process names are given in Table 5. *Unfold* and *Fold* express existence and uniqueness of a solution for the equation $\mathfrak{A}\stackrel{def}{=}t$, which correspond to Milner's standard axioms, and the *Recursive Definition Principle* (*RDP*) and *Recursive Specification Principle* (*RSP*) in *ACP*. *Unfold* states that each recursive operator has a solution (whether it is guarded or not), while *Fold* states that each guarded recursive operator has at most one solution.

Table 5. Axioms for process names.

$rec\mathfrak{A}\cdot t=t\{rec\mathfrak{A}\cdot t/\mathfrak{A}\}$	*Unfold*
$t_1=t_2\{t_1/\mathfrak{A}\}\Rightarrow t_1=rec\mathfrak{A}\cdot t_2$, if A is guarded in t_2	*Fold*
$rec\mathfrak{A}\cdot(\mathfrak{A}+t)=rec\mathfrak{A}\cdot t$	*Ung*
$rec\mathfrak{A}\cdot((\mathcal{C},\tau).((\mathcal{C}',\tau).t'+t)+s)=$ $rec\mathfrak{A}\cdot((\mathcal{C},\tau).(t'+t)+s)$, if $\mathfrak{A}$ is unguarded in t'	$WUng_1$
$rec\mathfrak{A}\cdot((\mathcal{C},\tau).(\mathfrak{A}+t)+s)=rec\mathfrak{A}\cdot((\mathcal{C},\tau).(t+s)+s)$	$WUng_2$
$\tau_m(rec\mathfrak{A}\cdot t)=rec\mathfrak{A}\cdot\tau_m(t)$, if $\mathfrak{A}$ is serial in t	*Hid*

The behavior of $\tau_{Msg}(\partial_{Msg}([\![P]\!]_A\parallel[\![Q]\!]_B))$ by using the axioms of Table 5 is expressed by:

$$\tau_{Msg}(\partial_{Msg}([\![P]\!]_B\parallel[\![Q]\!]_B))=\\ rec\mathfrak{Q}\cdot(\{A\rightsquigarrow B\},\tau).(\{\},deliver).\mathfrak{Q}+(\{A\not\rightsquigarrow B\},\tau).0$$

which explains that in case A is connected to B, each sending of $data_B$ is followed by the internal action $deliver$

It is not hard to see that the axioms of Table 3, Table 4 and Table 5 provide a sound axiomatization of *RCNT*. This can be checked by verifying soundness for each axiom individually.

Theorem 6.1. The axiomatization is sound, i.e. for all closed *RCNT* terms t_1 and t_2, if $t_1 = t_2$ then $t_1 \simeq_{rb} t_2$.

Our axiomatization is also ground-complete for terms with a finite-state CLTS, but not for infinite-state CLTSs. For example, $rec\mathfrak{W} \cdot (\{\}, nsnd(req, A)).\mathfrak{W} \parallel (\{? \rightsquigarrow B\}, nrcv(req)).\mathfrak{W}$ produces an infinite-state CLTS, since at each recursive call a new parallel operator is generated. Its equality to $rec\mathfrak{H} \cdot (\{\}, nsnd(req, A)).\mathfrak{H}$ cannot be proved by our axiomatization.

Theorem 6.2. The axiomatization is ground-complete, i.e., for all closed finite-state reliable computed network terms t_1 and t_2, $t_1 \simeq_{rb} t_2$ implies $t_1 = t_2$.

See sections C and D for the proofs of theorems 6.1 and 6.2, respectively.

7. Case study

In MANETs, nodes communicate through others via a multi-hop communication. Hence, nodes act as routers to make the communication possible among not directly connected nodes. We illustrate the applicability of our axioms in the analysis of MANET protocols through a simple routing protocol inspired by the AODV protocol.

7.1. Protocol specification

The protocol consists of three processes P, M, and Q, each specifying the behavior of a node as the source (that finds a route to a specific destination), middle node (that relays messages from the source to the destination), and destination. The description of these process are given in Figure 1.

Process P, deployed at the address A, uses the neighbor discovery service of the data link layer to examine if it has a direct link to the destination with the address B. If it is connected, then it sends its data directly by broadcasting the message $data_B$; otherwise, it initiates the route discovery procedure by sending the message req, then behaving as P_1. This process waits until it receives a reply from a middle name with the address C or B. In the former case, it behaves as P_2 which indicates that A sends it data through C as long as C is connected to A. In the latter case, it behaves as P which indicates that A sends it data as long as B is directly connected to A.

Process M relays req messages to find a route to B and then behaves as M_1. This process waits until it receives a reply. To model waits with a timeout, it non-deterministically sends a request again. Upon receiving a reply from C it behaves as M_2, indicating that it relays data messages of A as long as it has a link to C. Finally, process Q sends a reply upon receiving a request message and receives data messages.

$$
\begin{aligned}
P &\stackrel{def}{=} sense(B, snd(data_B).P, snd(req).P_1)\\
P_1 &\stackrel{def}{=} rcv(rep_C).P_2 + rcv(rep_B).P + snd(req).P_1\\
P_2 &\stackrel{def}{=} sense(C, rcv(error).P + snd(data_C).P_2, snd(req).P_1)\\
M &\stackrel{def}{=} rcv(req).snd(req).M_1\\
M_1 &\stackrel{def}{=} rcv(rep_B).snd(rep_C).M_2 + snd(req).M_1\\
M_2 &\stackrel{def}{=} sense(B, rcv(data_C).snd(data_B).M_2, snd(error).snd(req).M_1)\\
Q &\stackrel{def}{=} rcv(req).snd(rep_B).Q + rcv(data_B).deliver.Q
\end{aligned}
$$

Figure 1. The specification of processes P, M, and Q as a part of our simple routing protocol.

To simplify the route maintenance procedure of AODV, the middle node takes advantage of the sensing operator when it behaves as M_2. Whenever it finds out that it has no link to C, it sends an error message to its upstream node, i.e., A, to inform it that its route to B through C is not valid. After sending and receiving the error message, they both initiate the route discovery procedure by sending a request message.

The network with the three nodes of a source, middle, and destination is specified by

$$\mathcal{N} \equiv \tau_{Msg}(\partial_{Msg}(\llbracket P \rrbracket_A \parallel \llbracket M \rrbracket_C \parallel \llbracket Q \rrbracket_B)).$$

Analyzing $(\nu A)(\nu B)(\nu C)\mathcal{N}$, whose network addresses have been abstracted away, reveals that it is rooted branching bisimilar to a process with livelock behavior[1]. Possibly a livelock occurs where data is not delivered to B as no route can found to B. Such behavior may be the result of a conceptual mistake in the protocol design or the impossibility of communication between A and B due to their disconnectivity. We propose a technique in Section 7.2 to discover only those faulty behaviors that are due to an incorrect protocol design.

The network $\partial_{Msg}(\llbracket P \rrbracket_A \parallel \llbracket M \rrbracket_C \parallel \llbracket Q \rrbracket_B)$ can be simplified as:

$$
\begin{aligned}
&\partial_{Msg}(\llbracket P \rrbracket_A \parallel \llbracket M \rrbracket_C \parallel \llbracket Q \rrbracket_B) = \qquad (1)\\
&\quad (\{A \rightsquigarrow B\}, nsnd(data_B, A)).\partial_{Msg}(\llbracket P \rrbracket_A \parallel \llbracket M \rrbracket_C \parallel \llbracket deliver.Q \rrbracket_B) +\\
&\quad (\{A \not\rightsquigarrow B, A \rightsquigarrow C\}, nsnd(req, A)).\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket snd(req).M_1 \rrbracket_C \parallel \llbracket Q \rrbracket_B) +\\
&\quad (\{A \not\rightsquigarrow B, A \not\rightsquigarrow C\}, nsnd(req, A)).\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket M \rrbracket_C \parallel \llbracket Q \rrbracket_B).
\end{aligned}
$$

[1]The network has been specified in the mCRL2 language [28], following the approach of [18]. The code and its derived transition system modulo branching bisimilarity by the mCRL2 toolset is available at `fghassemi.adhoc.ir/codeFI17.zip`

Next, we simplify $\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket snd(req).M_1 \rrbracket_C \parallel \llbracket Q \rrbracket_B)$ as

$$\begin{aligned}\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket snd(req).M_1 \rrbracket_C \parallel \llbracket Q \rrbracket_B) = \\ (\{A \rightsquigarrow B\}, nsnd(req, A)).\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket snd(req).M_1 \rrbracket_C \parallel \llbracket snd(rep_B).Q \rrbracket_B) + \\ (\{A \not\rightsquigarrow B\}, nsnd(req, A)).\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket snd(req).M_1 \rrbracket_C \parallel \llbracket Q \rrbracket_B) + \\ (\{C \rightsquigarrow B\}, nsnd(req, C)).\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket M_1 \rrbracket_C \parallel \llbracket snd(rep_B).Q \rrbracket_B) + \\ (\{C \not\rightsquigarrow B\}, nsnd(req, C)).\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket M_1 \rrbracket_C \parallel \llbracket Q \rrbracket_B).\end{aligned} \tag{2}$$

Now, we continue by extending $\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket M_1 \rrbracket_C \parallel \llbracket snd(rep_B).Q \rrbracket_B)$:

$$\begin{aligned}\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket M_1 \rrbracket_C \parallel \llbracket snd(rep_B).Q \rrbracket_B) = \\ (\{\ \}, nsnd(req, A)).\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket M_1 \rrbracket_C \parallel \llbracket snd(rep_B).Q \rrbracket_B) + \\ (\{\ \}, nsnd(req, C)).\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket M_1 \rrbracket_C \parallel \llbracket snd(rep_B).Q \rrbracket_B) + \\ (\{B \rightsquigarrow A, C\}, nsnd(rep_B, B)).\partial_{Msg}(\llbracket P \rrbracket_A \parallel \llbracket snd(rep_C).M_2 \rrbracket_C \parallel \llbracket Q \rrbracket_B) + \\ (\{B \rightsquigarrow A, B \not\rightsquigarrow C\}, nsnd(rep_B, B)).\partial_{Msg}(\llbracket P \rrbracket_A \parallel \llbracket M_1 \rrbracket_C \parallel \llbracket Q \rrbracket_B) + \\ (\{B \not\rightsquigarrow A, B \rightsquigarrow C\}, nsnd(rep_B, B)).\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket snd(rep_C).M_2 \rrbracket_C \parallel \llbracket Q \rrbracket_B).\end{aligned}$$

By simplifying the term $\partial_{Msg}(\llbracket P \rrbracket_A \parallel \llbracket snd(rep_C).M_2 \rrbracket_C \parallel \llbracket Q \rrbracket_B)$, which indicates that A and C have found a direct route to B, we reach $\partial_{Msg}(\llbracket P \rrbracket_A \parallel \llbracket M_2 \rrbracket_C \parallel \llbracket Q \rrbracket_B)$:

$$\begin{aligned}\partial_{Msg}(\llbracket P \rrbracket_A \parallel \llbracket snd(rep_C).M_2 \rrbracket_C \parallel \llbracket Q \rrbracket_B) = \\ (\{\ \}, nsnd(rep_C, C)).\partial_{Msg}(\llbracket P \rrbracket_A \parallel \llbracket M_2 \rrbracket_C \parallel \llbracket Q \rrbracket_B) + \\ (\{A \rightsquigarrow B\}, nsnd(data_B, A)).\partial_{Msg}(\llbracket P \rrbracket_A \parallel \llbracket snd(rep_C).M_2 \rrbracket_C \parallel \llbracket deliver.Q \rrbracket_B) + \\ (\{A \not\rightsquigarrow B\}, nsnd(req, A)).\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket snd(rep_C).M_2 \rrbracket_C \parallel \llbracket Q \rrbracket_B).\end{aligned}$$

By extending $\partial_{Msg}(\llbracket P \rrbracket_A \parallel \llbracket M_2 \rrbracket_C \parallel \llbracket Q \rrbracket_B)$, we have:

$$\begin{aligned}\partial_{Msg}(\llbracket P \rrbracket_A \parallel \llbracket M_2 \rrbracket_C \parallel \llbracket Q \rrbracket_B) = \\ (\{A \rightsquigarrow B\}, nsnd(data_B, A)).\partial_{Msg}(\llbracket P \rrbracket_A \parallel \llbracket M_2 \rrbracket_C \parallel \llbracket deliver.Q \rrbracket_B) + \\ (\{A \not\rightsquigarrow B\}, nsnd(req, A)).\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket M_2 \rrbracket_C \parallel \llbracket Q \rrbracket_B).\end{aligned}$$

Finally extending $\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket M_2 \rrbracket_C \parallel \llbracket Q \rrbracket_B)$ results:

$$\begin{aligned}\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket M_2 \rrbracket_C \parallel \llbracket Q \rrbracket_B) = \\ (\{A \rightsquigarrow B\}, nsnd(req, A)).\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket M_2 \rrbracket_C \parallel \llbracket snd(rep_B).Q \rrbracket_B) + \\ (\{A \not\rightsquigarrow B\}, nsnd(req, A)).\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket M_2 \rrbracket_C \parallel \llbracket Q \rrbracket_B) + \\ (\{C \not\rightsquigarrow B\}, nsnd(error, C)).\partial_{Msg}(\llbracket P_1 \rrbracket_A \parallel \llbracket snd(req).M_1 \rrbracket_C \parallel \llbracket Q \rrbracket_B).\end{aligned}$$

The following scenario, found by above equations, is valid for a topology in which A has only a multi-hop link to B via C, but B has a direct link to A:

$$\begin{aligned}
&\partial_{Msg}(\llbracket P\rrbracket_A \parallel \llbracket M\rrbracket_C \parallel \llbracket Q\rrbracket_B)\\
&\xrightarrow{(\{A\not\rightsquigarrow B, A\rightsquigarrow C\}, nsnd(req,A))} \partial_{Msg}(\llbracket P_1\rrbracket_A \parallel \llbracket snd(req).M_1\rrbracket_C \parallel \llbracket Q\rrbracket_B)\\
&\xrightarrow{(\{C\rightsquigarrow B\}, nsnd(req,C))} \partial_{Msg}(\llbracket P_1\rrbracket_A \parallel \llbracket M_1\rrbracket_C \parallel \llbracket snd(rep_B).Q\rrbracket_B)\\
&\xrightarrow{(\{B\rightsquigarrow A, C\}, nsnd(rep_B,B))} \partial_{Msg}(\llbracket P\rrbracket_A \parallel \llbracket snd(rep_C).M_2\rrbracket_C \parallel \llbracket Q\rrbracket_B)\\
&\xrightarrow{(\{\ \}, nsnd(rep_C,C))} \partial_{Msg}(\llbracket P\rrbracket_A \parallel \llbracket M_2\rrbracket_C \parallel \llbracket Q\rrbracket_B)\\
&\xrightarrow{(\{A\not\rightsquigarrow B\}, nsnd(req,A))} \partial_{Msg}(\llbracket P_1\rrbracket_A \parallel \llbracket M_2\rrbracket_C \parallel \llbracket Q\rrbracket_B)\\
&\xrightarrow{(\{A\not\rightsquigarrow B\}, nsnd(req,A))} \partial_{Msg}(\llbracket P_1\rrbracket_A \parallel \llbracket M_2\rrbracket_C \parallel \llbracket Q\rrbracket_B)\\
&\ldots
\end{aligned}$$

The reason is found in the specification of M_2 which does not handle request messages, and hence, for such a topology no data will be received by B although there is a route from A to B and also a route from B to A. Therefore, we revise M_2 as:

$$\begin{aligned}
M_2 \ \stackrel{def}{=}\ & sense(B, rcv(data_C).snd(data_B).M_2 + rcv(req).snd(rep_C).M_2,\\
& snd(error).snd(req).M_1)
\end{aligned}$$

The path above also exists in the lossy setting, but with all disconnectivity pairs removed from the network constraints. However, an exhaustive and therefore expensive inspection of this path is needed to determine that it is due to a design error. The first transition in the path carries the label $(\{A \not\rightsquigarrow B, A \rightsquigarrow C\}, nsnd(req, A))$ in the reliable setting, meaning that B is not ready to receive, and the label $(\{A \rightsquigarrow C\}, nsnd(req, A))$ in the lossy setting. The latter label indicates that either B was not ready to receive or it was not connected to A. So in the lossy setting one has to examine the origin state to find out if B had an enabled receive action or not. The concept of not being ready to receive is treated in the same way as a lossy communication. Since only the former may be due to a conceptual design in the protocol, finding design errors is not straightforward in the lossy setting. In general the lossy setting will produce a large number of possible error traces that all need to be examined exhaustively, while the reliable setting will produce no spurious error traces.

7.2. Protocol analysis

The properties of wireless protocols, specially MANETs, tends to be weaker in comparison with the properties of wired protocols. For instance, the property of packet delivery from the node A to B is specified as "if there is a path from A to B for a long enough period of time, any packet sent by A, will be received by B" [16]. The topology-dependent behavior of communication, and consequently the need for multi-hop communication between nodes, make their properties preconditioned by the existence of some paths among nodes.

To investigate the topology-dependent properties of MANETs by equational reasoning, it is necessary to enrich our process theory *RCNT* to specify behaviors constrained by multi-hop constraints.

To this aim, we extend the action prefix operator of *RCNT* with actions that are paired with multi-hop constraints, first introduced in [18] and here extended by negative multi-hop connectivity pairs. Viewing a network topology as a directed graph, a multi-hop constraint is represented as a set of multi-hop (dis)connectivity pairs $\dashrightarrow$: $Loc \times Loc$ and $\not\dashrightarrow$: $Loc \times Loc$. For instance, $A \dashrightarrow C$ denotes there exists a multi-hop connection from A to C, and consequently C can indirectly receive data from A. Let $\mathbb{M}(Loc)$ denote the set of multi-hop constraints that can be defined over network addresses in Loc, ranged over by $\mathcal{M}$. Term $(\mathcal{M}, \iota).t$, where $\iota \in IAct \cup \{\tau\}$, denotes that the action ι is possible if the underlying topology satisfies the multi-hop network constraint $\mathcal{M}$. Formally, a topology like γ satisfies the multi-hop network constraint $\mathcal{M}$, denoted by $\gamma \models \mathcal{M}$ iff for each $\ell \dashrightarrow \ell'$ in $\mathcal{M}$, there is a multi-hop connection from ℓ to ℓ' in γ, and for each $\ell \not\dashrightarrow \ell'$ in $\mathcal{M}$, there is no multi-hop connection from ℓ to ℓ' in γ. To define a well-formed *RCNT* term, the rule which restricts the application of the new prefixed-actions to sequential processes, is added to the previous ones. Furthermore, a term cannot have two summands such that one is prefixed by an action of the form $(\mathcal{C}, \eta)$ and the other by an action of the form $(\mathcal{M}, \iota)$. So terms with an action of the form $(\mathcal{M}, \iota)$ only contain action prefix (with multi-hop constraints), choice and recursion operators.

To reason about the correctness of a MANET protocol, its behavior can be abstractly specified by observable internal actions with the required conditions on the underlying topology, i.e., ι-actions with multi-hop constraints. Intuitively, each communication of a protocol implementation triggers an internal action. Such communications are abstracted away by τ-transitions. Therefore, we define a novel preorder relation to examine if a protocol refines its specification. To this aim, a sequence of τ-transitions is allowed to precede an action that is matched to an action of the specification, as long as the accumulated network constraints of the τ-transitions satisfy the multi-hop network constraint of the matched action. By accumulating network constraints, it is ensured that the set of the links that make the multi-hop connections indicated by the multi-hop constraint hold, will be stable during the communications. Hence our preorder relation is parametrized by a network constraint to reflect such accumulated network constraints.

To provide such a relation, we use the notation $\stackrel{\mathcal{C}}{\Rightarrow}$ which is the reflexive and transitive closure of τ-relations while their network constraints are accumulated:

- $t \stackrel{\{\,\}}{\Longrightarrow} t$;
- if $t \xrightarrow{(\mathcal{C},\tau)} t'$ for some arbitrary network constraint $\mathcal{C}$ and $t' \stackrel{\mathcal{C}'}{\Rightarrow} t''$, then $t \stackrel{\mathcal{C}'\cup\mathcal{C}}{\Longrightarrow} t''$, where $\mathcal{C}' \cup \mathcal{C}$ is well-formed.

Furthermore, the network constraint $\mathcal{C}$ satisfies the multi-hop constraint $\mathcal{C}$, denoted by $\mathcal{C} \models \mathcal{M}$ iff $\exists\, \gamma \in \Gamma(\mathcal{C})\, (\gamma \models \mathcal{M})$. We remark that a network constraint like $\{A \not\rightsquigarrow B\}$ may satisfy both multi-hop constraints $\{A \dashrightarrow B\}$ and $\{A \not\dashrightarrow B\}$, but $\{A \rightsquigarrow B\}$ only satisfies $\{A \dashrightarrow B\}$.

Definition 7.1. A binary relation $\mathcal{R}_{\mathcal{C}}$ on $RCNT$ terms is a refinement relation if $t\, \mathcal{R}_{\mathcal{C}}\, s$ implies:

- if $t \xrightarrow{(\mathcal{C}',\eta)} t'$, where $\mathcal{C} \cup \mathcal{C}' \in \mathbb{C}^v(Loc)$, then
  - $\eta = \tau$ and $t'\, \mathcal{R}_{\mathcal{C}\cup\mathcal{C}'}\, s$, or

  - there is an s' such that $s \xrightarrow{(\mathcal{C},\eta)} s'$, and $t'\ \mathcal{R}_{\mathcal{C}\cup\mathcal{C}'}\ s'$, or
  - $\eta = \iota$ for some $\iota \in IAct \cup \{\tau\}$ and there is an s' such that $s \xrightarrow{(\mathcal{M},\iota)} s'$ with $\mathcal{C} \cup \mathcal{C}' \models \mathcal{M}$ and $t'\ \mathcal{R}_{\mathcal{C}\cup\mathcal{C}'}\ s'$;

- if $s \xrightarrow{(\mathcal{M},\iota)} s'$, then there are t'' and t' such that $t \overset{\mathcal{C}'}{\Longrightarrow} t'' \xrightarrow{(\mathcal{C}'',\iota)} t'$ with $\mathcal{C} \cup \mathcal{C}' \cup \mathcal{C}'' \models \mathcal{M}$ and $t'\ \mathcal{R}_{\mathcal{C}\cup\mathcal{C}'\cup\mathcal{C}''}\ s'$ where $\mathcal{C} \cup \mathcal{C}' \cup \mathcal{C}'' \in \mathbb{C}^v(Loc)$;

- if $s \xrightarrow{(\mathcal{C},\eta)} s'$, then there is a t' such that $t \xrightarrow{(\mathcal{C}',\eta)} t'$ with $t'\ \mathcal{R}_{\mathcal{C}\cup\mathcal{C}'}\ s'$.

The protocol t refines the specification s, denoted by $t \sqsubseteq s$, if $t\ \mathcal{R}_{\{\ \}}\ s$ holds for some refinement relation $\mathcal{R}_{\{\ \}}$.

Theorem 7.2. Refinement is a preorder relation and has the precongruence property.

See Section E for its proof. The following proposition is helpful to prove refinement between two processes:

Proposition 7.3. Suppose $\iota \in IAct$. The following rules holds

$$(\mathcal{C},\tau).t \sqsubseteq (\mathcal{M},\iota).s \Leftrightarrow \mathcal{C} \rhd t \sqsubseteq (\mathcal{M},\iota).s \wedge \mathcal{C} \models \mathcal{M}$$
$$(\mathcal{C},\iota).t \sqsubseteq (\mathcal{M},\iota).s \Leftrightarrow \mathcal{C} \rhd t \sqsubseteq s.$$

These rules correspond to the transfer conditions of Definition 7.1, and their proofs are discussed in Section E. Instead of finding a refinement relation and showing that it satisfies the transfer conditions of Definition 7.1 that its proof process is managed at the semantic level, we propose a proof process at the syntactic level. To show that $t \sqsubseteq s$, where s and t are closed terms for the specification and implementation, by our axiomatization, both s and t are rewritten into a set of summands like $(\mathcal{C}_1,\iota_1).t_1 + \ldots + (\mathcal{C}_n,\iota_n).t_n$ and $(\mathcal{M}_1,\iota'_1).s_1 + \ldots + (\mathcal{M}_m,\iota'_m).s_m$ for some $n \geq 0$ and $m \geq 0$. The proof of $t \sqsubseteq s$ can be reduced to proving that:

$$\forall i \leq n\,,\, \mathcal{C}_i \in \mathbb{C}^v(Loc)\,(\exists!\, j \leq m\,(\iota_i = \iota'_j \wedge \mathcal{C}_i \models \mathcal{M}_j \wedge \mathcal{C}_i \rhd t_i \sqsubseteq s_j) \vee (\mathcal{C}_i \rhd t_i \sqsubseteq s))$$

using the precongruence property of our refinement for the choice operator and the rules of Proposition 7.3. The first disjunct represents the third case of the first transfer condition while the second disjunct represents the first conditions of both transfer conditions of Definition 7.1. This proof process proceeds until we reach to a predicate $\mathcal{C}' \rhd t' \sqsubseteq s'$ to prove for which either we have previously examined $\mathcal{C}'' \rhd t' \sqsubseteq {}'s'$ where $\mathcal{C}' \preccurlyeq \mathcal{C}''$, or it holds trivially.

To analyze the correctness of our simple routing protocol, we investigate if it has the packet delivery property. To this end, we verify whether $\mathcal{N} \equiv \tau_{Msg}(\partial_{Msg}([\![P]\!]_A \parallel [\![M]\!]_C \parallel [\![Q]\!]_B))$ refines $\mathfrak{S}$, where $\mathfrak{S}$ is defined as:

$$\mathfrak{S} \overset{def}{=} (\{A \dashrightarrow B, B \dashrightarrow A\}, deliver).\mathfrak{S} + (\{A \not\dashrightarrow B\},\tau).0 + (\{A \dashrightarrow B, B \not\dashrightarrow A\},\tau).0.$$

Therefore, we exploit the provided equations in Section 7.1 to prove such a refinement at the syntactic level. To this aim, we match all the resulting terms of τ-transitions to $\mathfrak{S}$ as long as their accumulated

network constraints do not exclusively satisfy one of the multi-hop constraints $\{A \dashrightarrow B, B \dashrightarrow A\}$, $\{A \dashrightarrow B, B \not\dashrightarrow A\}$ or $\{A \not\dashrightarrow B\}$. Otherwise, if the accumulated network constraint of a τ-transition only satisfies $\{A \not\dashrightarrow B\}$ or $\{A \dashrightarrow B, B \not\dashrightarrow A\}$, the resulting term of the τ-transition will be matched to 0.

Thus, we use Equation 1 to show that:

$$\begin{aligned}
&\tau_{Msg}(\partial_{Msg}(\llbracket P\rrbracket_A \parallel \llbracket M\rrbracket_C \parallel \llbracket Q\rrbracket_B)) \sqsubseteq \mathfrak{S} \Leftrightarrow\\
&\{A \rightsquigarrow B\} \rhd \tau_{Msg}(\partial_{Msg}(\llbracket P\rrbracket_A \parallel \llbracket M\rrbracket_C \parallel \llbracket deliver.Q\rrbracket_B)) \sqsubseteq \mathfrak{S} \wedge\\
&\{A \not\rightsquigarrow B, A \rightsquigarrow C\} \rhd \tau_{Msg}(\partial_{Msg}(\llbracket P_1\rrbracket_A \parallel \llbracket snd(req).M_1\rrbracket_C \parallel \llbracket Q\rrbracket_B)) \sqsubseteq \mathfrak{S} \wedge\\
&\{A \not\rightsquigarrow B, A \not\rightsquigarrow C\} \rhd \tau_{Msg}(\partial_{Msg}(\llbracket P_1\rrbracket_A \parallel \llbracket M\rrbracket_C \parallel \llbracket Q\rrbracket_B)) \sqsubseteq 0
\end{aligned} \tag{3}$$

To prove the refinement relation 3, we use the Equation 2 to show that

$$\begin{aligned}
&\{A \not\rightsquigarrow B, A \rightsquigarrow C\} \rhd \tau_{Msg}(\partial_{Msg}(P_1\rrbracket_A \parallel \llbracket snd(req).M_1\rrbracket_C \parallel \llbracket Q\rrbracket_B)) \sqsubseteq \mathfrak{S} \Leftrightarrow\\
&\{A \not\rightsquigarrow B, A \rightsquigarrow C, C \rightsquigarrow B\} \rhd \tau_{Msg}(\partial_{Msg}(\llbracket P_1\rrbracket_A \parallel \llbracket M_1\rrbracket_C \parallel \llbracket snd(rep_B).Q\rrbracket_B)) \sqsubseteq \mathfrak{S} \wedge\\
&\{A \not\rightsquigarrow B, A \rightsquigarrow C, C \not\rightsquigarrow B\} \rhd \tau_{Msg}(\partial_{Msg}(\llbracket P_1\rrbracket_A \parallel \llbracket M_1\rrbracket_C \parallel \llbracket Q\rrbracket_B)) \sqsubseteq 0
\end{aligned} \tag{4}$$

The refinement relation (4) trivially holds as it can be proved with the help of our axiomatization, especially the rules *Fold* and $TRes_{1,2}$, that $\{A \not\rightsquigarrow B, A \rightsquigarrow C, C \not\rightsquigarrow B\} \rhd \tau_{Msg}(\partial_{Msg}(\llbracket P_1\rrbracket_A \parallel \llbracket M_1\rrbracket_C \parallel \llbracket Q\rrbracket_B))$ is the answer to the equation $\mathfrak{Q} \stackrel{def}{=} (\{A \not\rightsquigarrow B, A \rightsquigarrow C, C \not\rightsquigarrow B\}, \tau).\mathfrak{Q}$, and trivially

$$rec\mathfrak{Q} \cdot (\{A \not\rightsquigarrow B, A \rightsquigarrow C, C \not\rightsquigarrow B\}, \tau).\mathfrak{Q} \sqsubseteq 0.$$

So, it can be easily proved that $\tau_{Msg}(\partial_{Msg}(\llbracket P\rrbracket_A \parallel \llbracket M\rrbracket_C \parallel \llbracket Q\rrbracket_B)) \sqsubseteq \mathfrak{S}$.

8. Related work

Related frameworks to ours are CBS# [41], CWS [39], CMAN [24, 25], CMN [37] and its timed version [38], bKlaim [42], ω-calculus [47], SCWN [27], CSDT [34], AWN [16] and its timed extension [10], the broadcast psi-calculi [7] and wRebeca [51]. These approaches have already been compared in [21] with regard to modeling issues, such as topology and mobility, as well as behavioral congruence relations, in particular observables and distinguishing power. As all these approaches, except the approach of [39], focus on protocols above the data like layer, we investigate their capabilities to faithfully support the properties of wireless communication at this layer, i.e., being non-blocking and asynchronous. Recall that the non-blocking property ensures the sender broadcasts irrespective of the readiness of its receivers while asynchronicity makes the received packets be buffered at the receiver when it is busy with processing other messages. Furthermore, we compare our behavioral equivalence relation to those with a reliable setting.

All these approaches, except [51], provide an algebraic framework. Among them only [42] is intrinsically asynchronous, centered around the tuple space paradigm; broadcast messages are output into the tuple spaces of neighboring nodes to the sending node.

The non-blocking property is a consequence of either nodes being *input-enabled* or the communication primitives being lossy. In the former case, the asynchronous property is achieved through

abstract data specifications [13] in line with the approach from [29, 30], in which the sum operator plays a pivotal role. Each process is then parametrized by a variable of the queue type with a summand which receives all possible messages (if the queue is empty). Conversely, this property of the communication cannot be obtained in the frameworks with a lossy communication primitive such as CMN, CMAN, ω-calculus, SCWN, and the broadcast psi-calculi.

To make a process input-enabled while communication is synchronous, three approaches are followed. In the first approach, followed by AWN, the semantics is equipped with a rule similar to our rule Rcv_3 with a negative premise which expresses that if a node is not ready to receive, the message is simply ignored [16]. Due to our implicit modeling of topology, the negative premise of our rule is more complicated to characterize the unreadiness of nodes regarding the underlying topology. In the second approach, followed by CDST, counterparts for the rules Bro and $Recv$ are defined with negative premises to cover cases when a process cannot participate in the communication [34]. The third approach, provided by CSB#, eliminates negative premises, to remain within the de Simone format of structural operational semantics [16], in favor of actions which discard messages [41]. Therefore, the semantics is augmented by rules that trigger the ignore actions for any sending node, receiving nodes for disconnected locations, and deadlock. Furthermore, the rules Bro and $Recv$ are modified to cover cases when a process ignores a message.

Among the reliable settings, only CDST provides a behavioral equivalence relation, based on the notion of observational congruence: the receive and send actions are observable while transitions changing the underlying topology are treated as unobservable. However, due to implicit modeling of topology and mobility, our behavioral equivalence relation has been parametrized with network constraints while it considers the branching structure of MANETs.

9. Conclusion

We introduced the reliable framework *RRBPT*, suitable to specify and verify MANETs, with the aim to catch errors in design decisions. We discussed the required changes at the semantic model by extending the network constraints with negative connectivity links. Furthermore, we revised the equivalence relation of the lossy setting to preserve required behavior in the reliable framework. Then we demonstrated which axioms should be added to /removed from the reliable setting. We provided an analysis approach at the syntactic level, exploiting a precongruence relation and our axiomatization. We applied our analysis approach to a simple routing protocol to prove that it correctly finds routes among connected nodes.

References

[1] Baeten J, Bravetti M. A ground-complete axiomatization of finite state processes in process algebra, *Proc. 16th Conference on Concurrency Theory*, 3653, Springer, 2005, ISBN 3-540-28309-9.

[2] Basten T. Branching bisimilarity is an equivalence indeed!, *Information Processing Letters*, 1996; **58**(3):141–147. URL `https://doi.org/10.1016/0020-0190(96)00034-8`.

[3] Bengtson J, Johansson M, Parrow J, Victor B. Psi-calculi: Mobile processes, nominal data, and logic, *Proc. 24th Annual IEEE Symposium on Logic in Computer Science*, IEEE, 2009, ISBN 978-0-7695-3746-7.

[4] Bergstra J, Klop JW. Process algebra for synchronous communication, *Information and Control*, 1984;**60**(1-3):109–137. URL `https://doi.org/10.1016/S0019-9958(84)80025-X`.

[5] Bezem M, Groote JF. Invariants in process algebra with data, *Proc. 5th Conference of Concurrency Theory*, 836, Springer, 1994, ISBN 3-540-58329-7.

[6] Bhargavan K, Obradovid D, Gunter CA. Formal verification of standards for distance vector routing protocols, *Journal of the ACM*, 2002;**49**(4):538–576. doi:10.1145/581771.581775.

[7] Borgström J, Huang S, Johansson M, Raabjerg P, Victor B, Pohjola J Å, Parrow J. Broadcast psi-calculi with an application to wireless protocols, *Software and System Modeling*, 2015;**14**(1):201–216. doi:10.1007/s10270-013-0375-z.

[8] Bourke T, van Glabbeek RJ, Höfner P. A mechanized proof of loop freedom of the (untimed) AODV routing protocol, *Proc. 12th Symposium on Automated Technology for Verification and Analysis*, 8837, Springer, 2014, ISBN 978-3-319-11935-9.

[9] Bourke T, van Glabbeek RJ, Höfner P. Showing invariance compositionally for a process algebra for network protocols, *Proc. 5th Conference on Interactive Theorem Proving*, 8558, Springer, 2014, ISBN 978-3-319-08969-0.

[10] Bres, E., van Glabbeek, R. J., Höfner, P.: A timed process algebra for wireless networks with an application in routing (extended abstract), *Proc. 25th European Symposium on Programming*, 9632, Springer, 2016, ISBN 978-3-662-49497-4.

[11] Chiyangwa S, Kwiatkowska M. A timing analysis of AODV, *Proc. 7th IFIP Conference on Formal Methods for Open Object-based Distributed Systems*, 3535, Springer, 2005, ISBN 3-540-26181-8.

[12] Clarke EM, Emerson EA. Design and synthesis of synchronization skeletons using branching-time temporal logic, *Proc. Workshop on Logic of Programs*, 131, Springer, 1981, ISBN 3-540-11212-X.

[13] Ehrich H, Loeckx J, Wolf M. *Specification of Abstract Data Types*, John Wiley, 1996, ISBN 978-0-471-95067-7.

[14] Fall KR, Stevens WR. *TCP/IP Illustrated*, vol. 1, Addison-Wesley, 2011, ISBN 0-13-280818-8.

[15] Fehnker A, van Glabbeek RJ, Höfner P, McIver A, Portmann, M, Tan WL. Automated analysis of AODV using UPPAAL, *Proc. 18th Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 7214, Springer, 2012, ISBN 978-3-642-28755-8.

[16] Fehnker A, van Glabbeek RJ, Höfner P, McIver A, Portmann M, Tan WL. A process algebra for wireless mesh networks, *Proc. 21st European Symposium on Programming*, 7211, Springer, 2012, ISBN 978-3-642-28868-5.

[17] Fokkink WJ, Pang J, van de Pol JC. Cones and foci: A mechanical framework for protocol verification, *Formal Methods in System Design*, 2006;**29**(1):1–31. doi:10.1007/s10703-006-0004-3.

[18] Ghassemi F, Fokkink WJ. Model checking mobile ad hoc networks, *Formal Methods in System Design*, 2016;**48**(1-2):159–189. doi:10.1007/s10703-016-0254-7.

[19] Ghassemi F, Fokkink WJ, Movaghar A. Restricted broadcast process theory, *Proc. 6th Conference on Software Engineering and Formal Methods*, IEEE, 2008, ISBN 978-0-7695-3437-4.

[20] Ghassemi F, Fokkink WJ, Movaghar A. Equational reasoning on ad hoc networks, *Proc. 3rd Conference on Fundamentals of Software Engineering*, 5961, Springer, 2009, ISBN 978-3-642-11622-3.

[21] Ghassemi F, Fokkink WJ, Movaghar A. Equational reasoning on mobile ad hoc networks, *Fundamenta Informaticae*, 2010;**105**(4):375–415.

[22] Ghassemi F, Fokkink WJ, Movaghar A. Verification of mobile ad hoc networks: An algebraic approach, *Theoretical Computer Science*, 2011;**412**(28):3262–3282. URL `https://doi.org/10.1016/j.tcs.2011.03.017`.

[23] van Glabbeek RJ, Höfner P, Portmann M, Tan WL. Modelling and verifying the AODV routing protocol, *Distributed Computing*, 2016;**29**(4):279–315. doi:10.1007/s00446-015-0262-7.

[24] Godskesen J. A calculus for mobile ad hoc networks, *Proc. 9th Conference on Coordination Models and Languages*, 4467, Springer, 2007, ISBN 978-3-540-72793-4.

[25] Godskesen J. A calculus for mobile ad-hoc networks with static location binding, *Proc. 15th Workshop on Expressiveness in Concurrency*, 2009;242(1):161–183. doi:10.1016/j.entcs.2009.06.018.

[26] Godskesen J, Gryn O. Modeling and verification of security protocols for ad hoc networks using UPPAAL, *Proc. 18-th Nordic Workshop on Programming Theory*, 2006.

[27] Godskesen J, Nanz S. Mobility models and behavioural equivalence for wireless networks, *Proc. 11th Conference on Coordination Models and Languages*, 5521, Springer, 2009, ISBN 978-3-642-02052-0.

[28] Groote JF, Mathijssen A, Reniers M, Usenko Y, van Weerdenburg M. The formal specification language mCRL2, *Methods for Modelling Software Systems*, 06351, Schloss Dagstuhl, 2006, ISSN 1862-4405.

[29] Groote JF, Ponse A. μCRL: A base for analysing processes with data, *Proc. 3rd Workshop on Concurrency and Compositionality*, GMD-Studien Nr. 191, 1991.

[30] Groote JF, Ponse A. Syntax and semantics of μCRL, *Proc. Workshop on Algebra of Communicating Processes*, Workshops in Computing, Springer, 1995, ISBN 978-3-540-19909-0.

[31] Groote JF, Springintveld J. Focus points and convergent process operators: A proof strategy for protocol verification, *Journal of Logic and Algebraic Programming*, 2001;**49**(1-2):31–60. URL `https://doi.org/10.1016/S1567-8326(01)00010-8`.

[32] Groote JF, van Wamel J. The parallel composition of uniform processes with data, *Theoretical Computer Science*, 2001;**266**(1-2):631–652. URL `https://doi.org/10.1016/S0304-3975(00)00324-8`.

[33] Höfner P, van Glabbeek RJ, Tan WL, Portmann M, McIver A, Fehnker A. A rigorous analysis of AODV and its variants, *The 15th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ACM, 2012, ISBN 978-1-4503-1628-6.

[34] Kouzapas D, Philippou A. A process calculus for dynamic networks, *Proc. Conference on Formal Techniques for Distributed Systems*, 6722, Springer, 2011, ISBN 978-3-642-21460-8.

[35] Luttik B. *https://doi.org/10.1016/S0304-3975(00)00324-8*, Ph.D. Thesis, University of Amsterdam, 2002.

[36] McIver A, Fehnker A. Formal techniques for analysis of wireless networks, *Proc. 2nd Symposium on Leveraging Applications of Formal Methods*, IEEE, 2006, ISBN 978-0-7695-3071-0.

[37] Merro M. An observational theory for mobile ad hoc networks, *Information and Computation*, 2009;**207**(2):194–208. URL `https://doi.org/10.1016/j.ic.2007.11.010`.

[38] Merro M, Ballardin F, Sibilio EA. A timed calculus for wireless systems, *Theoretical Computer Science*, 2011;**412**(47):6585–6611. URL `https://doi.org/10.1016/j.tcs.2011.07.016`.

[39] Mezzetti N, Sangiorgi D. Towards a calculus for wireless systems, *Proc. 22nd Conference on Mathematical Foundations of Programming Semantics*, 2006;**158**:331–353. URL `https://doi.org/10.1016/j.entcs.2006.04.017`.

[40] Namjoshi KS, Trefler RJ. Loop freedom in AODVv2, *Proc. 35th IFIP Conference on Formal Techniques for Distributed Objects, Components, and Systems*, 9039, 2015, ISBN 978-3-319-19194-2.

[41] Nanz S, Hankin C. A framework for security analysis of mobile wireless networks, *Theoretical Computer Science*, 2006;**367**(1-2):203–227.
URL `Aframeworkforsecurityanalysisofmobilewirelessnetworks,`.

[42] Nanz S, Nielson F, Nielson H. Static analysis of topology-dependent broadcast networks, *Information and Computation*, 2010;**208**(2):117–139. doi:10.1016/j.ic.2009.10.003.

[43] Perkins CE, Belding-Royer EM. Ad-hoc on-demand distance vector routing, *Proc. 2nd Workshop on Mobile Computing Systems and Applications*, IEEE, 1999, ISBN 0-7695-0025-0.

[44] Plotkin GD. A structural approach to operational semantics, *Journal of Logic and Algebraic Programming*, 2004;**60-61**:17–139. doi:10.1016/j.jlap.2004.05.001.

[45] Prasad KVS. A calculus of broadcasting systems, *Science of Computer Programming*, 1995;**25**(2-3):285–327. URL `https://doi.org/10.1016/0167-6423(95)00017-8`.

[46] de Renesse R, Aghvami AH. Formal verification of ad-hoc routing protocols using SPIN model checker, *Proc. 12th Mediterranean Electrotechnical Conference*, IEEE, 2004, ISBN 0-7803-8271-4.

[47] Singh A, Ramakrishnan CR, Smolka SA. A process calculus for mobile ad hoc networks, *Science of Computer Programming*, 2010;**75**(6):440–469. URL `https://doi.org/10.1016/j.scico.2009.07.008`.

[48] Usenko Y. *Linearization in μCRL*, Ph.D. Thesis, Eindhoven University of Technology, 2002.

[49] Wibling O, Parrow J, Pears A. Automatized verification of ad hoc routing protocols, *Proc. 24th IFIP Conference on Formal Techniques for Networked and Distributed Systems*, 3235, Springer, 2004. doi:10.1007/978-3-540-30232-2_22.

[50] Wibling O, Parrow J, Pears A. Ad hoc routing protocol verification through broadcast abstraction, *Proc. 25th IFIP Conference on Formal Techniques for Networked and Distributed Systems*, 3731, Springer, 2005, ISBN 3-540-29189-X.

[51] Yousefi B, Ghassemi F, Khosravi R. Modeling and efficient verification of wireless ad hoc networks, *Formal Aspect of Computing*, 2017;**29**(6):1051–1086. doi:10.1007/s00165-017-0429-z.

A. Branching reliable computed network bisimilarity is an equivalence

To prove that branching reliable computed network bisimilarity is an equivalence, we exploit semi-branching reliable computed network bisimilarity, following [2].

Definition A.1. A binary relation $\mathcal{R}$ on computed network terms is a semi-branching reliable computed network simulation, if $t_1 \mathcal{R} t_2$ implies whenever $t_1 \xrightarrow{(C,\eta)} t_1'$:

- either $\eta = \tau$ and there is a t_2' such that $t_2 \Rightarrow t_2'$ with $t_1 \mathcal{R} t_2'$ and $t_1' \mathcal{R} t_2'$; or
- there are $s_1'', \ldots, s_k''$ and $s_1', \ldots, s_k'$ for some $k > 0$ such that $\forall i \leq k\, (t_2 \Rightarrow s_i'' \xrightarrow{\langle (\mathcal{C}_i, \eta) \rangle} s_i'$, with $t_1 \mathcal{R} s_i''$ and $t_1' \mathcal{R} s_i')$, and $\langle \mathcal{C}_1 \rangle, \ldots, \langle \mathcal{C}_k \rangle$ constitute a partitioning of $\langle \mathcal{C} \rangle$.

$\mathcal{R}$ is a semi-branching reliable computed network bisimulation if $\mathcal{R}$ and $\mathcal{R}^{-1}$ are semi-branching reliable computed network simulations. Computed networks t_1 and t_2 are semi-branching reliable computed network bisimilar if $t_1 \mathcal{R} t_2$, for some semi-branching reliable computed network bisimulation relation $\mathcal{R}$.

Lemma A.2. Let t_1 and t_2 be computed network terms, and $\mathcal{R}$ a semi-branching reliable computed network bisimulation such that $t_1 \mathcal{R} t_2$.

- If $t_1 \Rightarrow t_1'$ then $\exists t_2' \cdot t_2 \Rightarrow t_2' \wedge t_1' \mathcal{R} t_2'$
- If $t_2 \Rightarrow t_2'$ then $\exists t_1' \cdot t_1 \Rightarrow t_1' \wedge t_1' \mathcal{R} t_2'$

Proof:
We only give the proof of the first property. The second property can be proved in a similar fashion. The proof is by induction on the number of $\Rightarrow$ steps from t_1 to t_1':

- Base: Assume that the number of steps equals zero. Then t_1 and t_1' must be equal. Since $t_1 \mathcal{R} t_2$ and $t_2 \Rightarrow t_2$, the property is satisfied.
- Induction step: Assume $t_1 \Rightarrow t_1'$ in n steps, for some $n \geq 1$. Then there is t_1'' such that $t_1 \Rightarrow t_1''$ in $n-1 \Rightarrow$ steps, and $t_1'' \xrightarrow{(\mathcal{C}, \tau)} t_1'$. By the induction hypothesis, there is a t_2'' such that $t_2 \Rightarrow t_2''$ and $t_1'' \mathcal{R} t_2''$. Since $t_1'' \xrightarrow{(\mathcal{C}, \tau)} t_1'$ and $\mathcal{R}$ is a semi-branching reliable computed network bisimulation, there are two cases to consider:
  - there is a t_2' such that $t_2'' \Rightarrow t_2'$, $t_1'' \mathcal{R} t_2'$, and $t_1' \mathcal{R} t_2'$. So $t_2 \Rightarrow t_2'$ such that $t_1' \mathcal{R} t_2'$.
  - or there are $s_1''', \ldots, s_k'''$ and $s_1', \ldots, s_k'$ for some $k > 0$ such that $\forall i \leq k\, (t_2'' \Rightarrow s_i''' \xrightarrow{(\mathcal{C}_i, \tau)} s_i'$, with $t_1'' \mathcal{R} s_i'''$ and $t_1' \mathcal{R} s_i')$, and $\mathcal{C}_1, \ldots, \mathcal{C}_k$ constitute a partitioning of $\mathcal{C}$. By definition, $s_i''' \xrightarrow{(\mathcal{C}_i, \tau)} s_i'$ yields $s_i''' \Rightarrow s_i'$. Consequently for any arbitrary $i \leq k$, $t_2 \Rightarrow s_i'$ such that the relation $t_1' \mathcal{R} s_i'$ holds.

□

Proposition A.3. The relation composition of two semi-branching reliable computed network bisimulations is again a semi-branching reliable computed network bisimulation.

Proof:
Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be semi-branching reliable computed network bisimulations with $t_1 \mathcal{R}_1 t_2$ and $t_2 \mathcal{R}_2 t_3$. Let $t_1 \xrightarrow{(\mathcal{C}, \eta)} t_1'$. It must be shown that

- either $\eta = \tau$ and there is a t_3' such that $t_3 \Rightarrow t_3'$ with $t_1 \mathcal{R}_1 \circ \mathcal{R}_2 t_3'$ and $t_1' \mathcal{R}_1 \circ \mathcal{R}_2 t_3'$; or

- $\exists s'_1,\ldots,s'_k,s''_1,\ldots,s''_k\,\forall i\le k\,(t_3\Rightarrow s''_i \xrightarrow{\langle(\mathcal{C}_i,\eta)\rangle} s'_i \wedge t_1\mathcal{R}_1\circ\mathcal{R}_2 s''_i \wedge t'_1\mathcal{R}_1\circ\mathcal{R}_2 s'_i)$, where $\langle\mathcal{C}_1\rangle,\ldots,\langle\mathcal{C}_k\rangle$ constitute a partitioning of $\langle\mathcal{C}\rangle$.

Since $t_1\mathcal{R}_1 t_2$, two cases can be considered:

- $\eta=\tau$ and there is a t'_2 such that $t_2\Rightarrow t'_2$ with $t_1\mathcal{R}_1 t'_2$ and $t'_1\mathcal{R}_1 t'_2$. Lemma A.2 yields that there is a t'_3 that $t_3\Rightarrow t'_3$ with $t'_2\mathcal{R}_2 t'_3$. It immediately follows that $t_1\mathcal{R}_1\circ\mathcal{R}_2 t'_3$ and $t'_1\mathcal{R}_1\circ\mathcal{R}_2 t'_3$.

- there exist $s^{**}_1,\ldots s^{**}_j$, $s^{*}_1\ldots s^{*}_j$ for some $j>0$ such that $\forall i\le j\,(t_2\Rightarrow s^{**}_i \xrightarrow{\langle(\mathcal{C}_i,\eta)\rangle} s^{*}_i$, $t_1\mathcal{R}_1 s^{**}_i$, $t'_1\mathcal{R}_1 s^{*}_i)$, and $\langle\mathcal{C}_1\rangle,\ldots,\langle\mathcal{C}_j\rangle$ is a partitioning of $\langle\mathcal{C}\rangle$. Since $t_2\mathcal{R}_2 t_3$ and $t_2\Rightarrow s^{**}_i$, Lemma A.2 yields that there are $s'''_1,\ldots,s'''_j$ such that $\forall i\le j\,(t_3\Rightarrow s'''_i\wedge s^{**}_i\mathcal{R}_2 s'''_i)$. Two cases can be distinguished:

  - either $\eta=\tau$ and for some $i\le j$, $s^{**}_i \xrightarrow{(\mathcal{C}_i,\tau)} s^{*}_i$ implies there is an s''_i such that $s'''_i\Rightarrow s''_i$ with $s^{**}_i\mathcal{R}_2 s''_i$ and $s^{*}_i\mathcal{R}_2 s''_i$. It follows immediately that there is an s''_i such that $t_3\Rightarrow s''_i$ with $t_1\mathcal{R}_1\circ\mathcal{R}_2 s''_i$ and $t'_1\mathcal{R}_1\circ\mathcal{R}_2 s''_i$; or
  - for all $i\le j$, $s^{**}_i \xrightarrow{\langle(\mathcal{C}_i,\eta)\rangle} s^{*}_i$ implies there are $s''_{i_1},\ldots,s''_{i_{k_i}}$ and $s'_{i_1},\ldots,s'_{i_{k_i}}$ for some $k_i>0$ such that $\forall o\le k_i\,(s''_i\Rightarrow s''_{i_o} \xrightarrow{\langle(\mathcal{C}_{i_o},\eta)\rangle} s'_{i_o}$, $s^{**}_i\mathcal{R}_2 s''_{i_o}$, $s^{*}_i\mathcal{R}_2 s'_{i_o})$, and $\langle\mathcal{C}_{i_1}\rangle,\ldots,\langle\mathcal{C}_{i_{k_i}}\rangle$ is a partitioning of $\langle\mathcal{C}_i\rangle$. Since $t_3\Rightarrow s''_i$, we have $\forall i\le j$, $\forall o\le k_i\,(t_3\Rightarrow s''_{i_o} \xrightarrow{\langle(\mathcal{C}_{i_o},\eta)\rangle} s'_{i_o}$ with $t_1\mathcal{R}_1\circ\mathcal{R}_2 s''_{i_o}$, $t'_1\mathcal{R}_1\circ\mathcal{R}_2 s'_{i_o})$, and $\{\langle\mathcal{C}_{i_o}\rangle \mid i\le j, o<k_i\}$ is a partitioning of $\langle\mathcal{C}\rangle$.

□

Corollary A.4. Semi-branching reliable computed network bisimilarity is an equivalence relation.

It is not hard to see that the union of semi-branching reliable computed network bisimulations is again a semi-branching reliable computed network bisimulation.

Proposition A.5. The largest semi-branching reliable computed network bisimulation is a branching reliable computed network bisimulation.

Proof:
Suppose $\mathcal{R}$ is the largest semi-branching reliable computed network bisimulation for some given CLTS. Let $t_1\mathcal{R}t_2$, $t_2\Rightarrow t'_2$, $t_1\mathcal{R}t'_2$ and $t'_1\mathcal{R}t'_2$. We show that $\mathcal{R}'=\mathcal{R}\cup\{(t'_1,t_2)\}$ is a semi-branching reliable computed network bisimulation.

1. If $t'_1 \xrightarrow{(\mathcal{C},\eta)} t''_1$, then it follows from $(t'_1,t'_2)\in\mathcal{R}$ that

   - either $\eta=\tau$ and there is a t''_2 such that $t'_2\Rightarrow t''_2$ with $t'_1\mathcal{R}t''_2$ and $t''_1\mathcal{R}t''_2$. Finally $t_2\Rightarrow t'_2$ results $t'_1\mathcal{R}'t''_2$ and $t''_1\mathcal{R}'t''_2$; or
   - there are $s'''_1,\ldots,s'''_k$ and $s''_1,\ldots,s''_k$ for some $k>0$ such that $\forall i\le k\,(t'_2\Rightarrow s'''_i \xrightarrow{\langle(\mathcal{C}_i,\eta)\rangle} s''_i$ with $(t'_1,s'''_i),(t''_1,s''_i)\in\mathcal{R})$ and $\langle\mathcal{C}_1\rangle,\ldots,\langle\mathcal{C}_k\rangle$ is a partitioning of $\langle\mathcal{C}\rangle$. And $t_2\Rightarrow t'_2$ yields $\forall i\le k\,(t_2\Rightarrow s'''_i \xrightarrow{\langle(\mathcal{C}_i,\eta)\rangle} s''_i$, with $(t'_1,s'''_i),(t''_1,s''_i)\in\mathcal{R}')$.

2. If $t_2 \xrightarrow{(\mathcal{C},\eta)} t_2''$, then it follows from $(t_1, t_2) \in \mathcal{R}$ that

   - either $\eta = \tau$, and there is a t_1'' such that $t_1 \Rightarrow t_1''$ with $t_1'' \mathcal{R} t_2$ and $t_1'' \mathcal{R} t_2''$. Furthermore, $(t_1, t_2') \in \mathcal{R}$, $t_1 \Rightarrow t_1''$, and Lemma A.2 imply there is a t_2''' such that $t_2' \Rightarrow t_2'''$ with $(t_1'', t_2''') \in \mathcal{R}$. Similarly $(t_1', t_2') \in \mathcal{R}$, $t_2' \Rightarrow t_2'''$, and Lemma A.2 imply there is a t_1''' such that $t_1' \Rightarrow t_1'''$ with $(t_1''', t_2''') \in \mathcal{R}$. From $(t_1''', t_2''') \in \mathcal{R}$, $(t_2''', t_1'') \in \mathcal{R}^{-1}$, and $(t_1'', t_2) \in \mathcal{R}$, we conclude $(t_1''', t_2) \in \mathcal{R} \circ \mathcal{R}^{-1} \circ \mathcal{R}$. And from $(t_1''', t_2''') \in \mathcal{R}$, $(t_2''', t_1'') \in \mathcal{R}^{-1}$, and $(t_1'', t_2'') \in \mathcal{R}$, we conclude $(t_1''', t_2'') \in \mathcal{R} \circ \mathcal{R}^{-1} \circ \mathcal{R}$.
   - or there are $s_{1_1}''', \ldots, s_{1_k}'''$ and $s_{1_1}'', \ldots, s_{1_k}''$ for some $k > 0$ such that $\forall i \leq k\,(t_1 \Rightarrow s_{1_i}''' \xrightarrow{\langle(\mathcal{C}_i,\eta)\rangle} s_{1_i}''$ with $(s_{1_i}''', t_2), (s_{1_i}'', t_2'') \in \mathcal{R})$ and $\langle\mathcal{C}_1\rangle, \ldots, \langle\mathcal{C}_k\rangle$ is a partitioning of $\langle\mathcal{C}\rangle$. Since $(t_1, t_2') \in \mathcal{R}$ and $t_1 \Rightarrow s_{1_i}'''$, by Lemma A.2, there are $s_{2_1}''', \ldots, s_{2_k}'''$ such that $\forall i \leq k\,(t_2' \Rightarrow s_{2_i}'''$ and $(s_{1_i}''', s_{2_i}''') \in \mathcal{R})$. Since $s_{1_i}''' \xrightarrow{\langle(\mathcal{C}_i,\eta)\rangle} s_{1_i}''$, there are $s_{2_{i_1}}^{**}, \ldots, s_{2_{i_{k_i}}}^{**}$ and $s_{2_{i_1}}^{*}, \ldots, s_{2_{i_{k_i}}}^{*}$ for some $k_i > 0$ such that $\forall o \leq k_i\,(s_{2_i}''' \Rightarrow s_{2_{i_o}}^{**} \xrightarrow{\langle(\mathcal{C}_{i_o},\eta)\rangle} s_{2_{i_o}}^{*}$ with $(s_{1_i}''', s_{2_{i_o}}^{**}), (s_{1_i}'', s_{2_{i_o}}^{*}) \in \mathcal{R})$ and $\langle\mathcal{C}_{i_1}\rangle, \ldots, \langle\mathcal{C}_{i_{k_i}}\rangle$ is a partitioning of $\langle\mathcal{C}_i\rangle$. Since $t_2' \Rightarrow s_{2_i}'''$ and $s_{2_i}''' \Rightarrow s_{2_{i_o}}^{**}$, we have $\forall i \leq k, o \leq k_i\,(t_2' \Rightarrow s_{2_{i_o}}^{**})$. By assumption, $(t_1', t_2') \in \mathcal{R}$, so by Lemma A.2 there are $s_{1_1}^{**}, \ldots, s_{1_K}^{**}$, where $K = \sum_{i=1}^{k} k_i$, such that $\forall z \leq K\,(t_1' \Rightarrow s_{1_z}^{**}$ and $(s_{1_z}^{**}, s_{2_{i_o}}^{**}) \in \mathcal{R}$, where $z = (\sum_{j=1}^{i-1} k_j) + o)$. Since $s_{2_{i_o}}^{**} \xrightarrow{\langle(\mathcal{C}_{i_o},\eta)\rangle} s_{2_{i_o}}^{*}$, there are $s_{1_{z_1}}^{***}, \ldots, s_{1_{z_{k_z'}}}^{***}$ and $s_{1_{z_1}}^{'}, \ldots, s_{1_{z_{k_z'}}}^{'}$ for some $k_z' > 0$ such that $\forall j \leq k_z'\,(s_{1_z}^{**} \Rightarrow s_{1_{z_j}}^{***} \xrightarrow{\langle(\mathcal{C}_{i_{o_j}},\eta)\rangle} s_{1_{z_j}}^{'}$ with $(s_{1_{z_j}}^{***}, s_{2_{i_o}}^{**}), (s_{1_{z_j}}^{'}, s_{2_{i_o}}^{*}) \in \mathcal{R})$ and $\langle\mathcal{C}_{i_{o_1}}\rangle, \ldots, \langle\mathcal{C}_{i_{o_{k_z'}}}\rangle$ is a partitioning of $\langle\mathcal{C}_{i_o}\rangle$. And $t_1' \Rightarrow s_{1_z}^{**}$ yields $\forall i \leq k, o \leq k_i, j \leq k_z'(t_1' \Rightarrow s_{1_{z_j}}^{***} \xrightarrow{\langle(\mathcal{C}_{i_{o_j}},\eta)\rangle} s_{1_{z_j}}^{'}$ with

   $$\begin{aligned}(s_{1_{z_j}}^{***}, s_{2_{i_o}}^{**}) \in \mathcal{R} \wedge (s_{2_{i_o}}^{**}, s_{1_i}''') \in \mathcal{R}^{-1} \wedge (s_{1_i}''', t_2) \in \mathcal{R} \\ \Rightarrow (s_{1_{z_j}}^{***}, t_2) \in \mathcal{R} \circ \mathcal{R}^{-1} \circ \mathcal{R} \\ (s_{1_{z_j}}^{'}, s_{2_{i_o}}^{*}) \in \mathcal{R} \wedge (s_{2_{i_o}}^{*}, s_{1_i}'') \in \mathcal{R}^{-1} \wedge (s_{1_i}'', t_1'') \in \mathcal{R} \\ \Rightarrow (s_{1_{z_j}}^{'}, t_2'') \in \mathcal{R} \circ \mathcal{R}^{-1} \circ \mathcal{R},\end{aligned}$$

   where $z = (\sum_{l=1}^{i-1} k_l) + o)$, and $\{\langle\mathcal{C}_{i_{o_j}}\rangle \mid i \leq k, o \leq k_i, j \leq k_z'\}$ is a partitioning of $\langle\mathcal{C}\rangle$.

   By Proposition A.3, $\mathcal{R} \circ \mathcal{R}^{-1} \circ \mathcal{R}$ is a semi-branching reliable computed network bisimulation. Since $\mathcal{R}$ is the largest semi-branching reliable computed network bisimulation, and clearly $\mathcal{R} \subseteq \mathcal{R} \circ \mathcal{R}^{-1} \circ \mathcal{R}$, we have $\mathcal{R} = \mathcal{R} \circ \mathcal{R}^{-1} \circ \mathcal{R}$.

So $\mathcal{R}'$ is a semi-branching reliable computed network bisimulation. Since $\mathcal{R}$ is the largest semi-branching reliable computed network bisimulation, $\mathcal{R}' = \mathcal{R}$.

We will now prove that $\mathcal{R}$ is a branching reliable computed network bisimulation. Let $t_1 \mathcal{R} t_2$, and $t_1 \xrightarrow{(\mathcal{C},\eta)} t_1'$. We only consider the case when $\eta = \tau$, because for other cases, the transfer condition of Definition 5.3 and Definition A.1 are the same. Two cases can be distinguished:

1. There is a t_2' such that $t_2 \Rightarrow t_2'$ with $t_1 \mathcal{R} t_2'$ and $t_1' \mathcal{R} t_2'$: we proved above that $t_1' \mathcal{R} t_2$. This agrees with the first case of Definition 5.3.

2. There are $s_1'', \ldots, s_k''$ and $s_1', \ldots, s_k'$ for some $k > 0$ such that $\forall i \leq k\,(t_2 \Rightarrow s_i'' \xrightarrow{\langle (\mathcal{C}_i, \tau) \rangle} s_i'$ with $t_1 \mathcal{R} s_i''$ and $t_1' \mathcal{R} s_i'$) and $\langle \mathcal{C}_1 \rangle, \ldots, \langle \mathcal{C}_k \rangle$ constitute a partitioning of $\langle \mathcal{C} \rangle$. This agrees with the second case of Definition 5.1.

Consequently $\mathcal{R}$ is a branching reliable computed network bisimulation. □

Since any branching reliable computed network bisimulation is a semi-branching reliable computed network bisimulation, this yields the following corollary.

Corollary A.6. Two computed network terms are related by a branching reliable computed network bisimulation if and only if they are related by a semi-branching reliable computed network bisimulation.

Corollary A.7. Branching reliable computed network bisimilarity is an equivalence relation.

Corollary A.8. Rooted branching reliable computed network bisimilarity is an equivalence relation.

Proof:
It is easy to show that rooted branching reliable computed network bisimilarity is reflexive and symmetric. To conclude the proof, we show that rooted branching reliable computed network bisimilarity is transitive. Let $t_1 \simeq_{rbr} t_2$ and $t_2 \simeq_{rbr} t_3$. Since $t_1 \simeq_{rbr} t_2$, if $t_1 \xrightarrow{(\mathcal{C},\eta)} t_1'$, then there is t_2' such that $t_2 \xrightarrow{\langle (\mathcal{C},\eta) \rangle} t_2'$ and $t_1' \simeq_{br} t_2'$. Since $t_2 \simeq_{rbr} t_3$, there is a t_3' such that $t_3 \xrightarrow{\langle (\mathcal{C},\eta) \rangle} t_3'$ and $t_2' \simeq_{br} t_3'$. Equivalence of branching reliable computed network bisimilarity yields $t_3 \xrightarrow{\langle (\mathcal{C},\eta) \rangle} t_3'$ with $t_1' \simeq_{br} t_3'$. The same argumentation holds when $t_3 \xrightarrow{(\mathcal{C},\eta)} t_3'$. Consequently the transfer conditions of Definition 5.5 holds and $t_1 \simeq_{rbr} t_3$. □

B. Rooted branching reliable computed network bisimilarity is a congruence

Theorem B.1. Rooted branching reliable computed network bisimilarity is a congruence for terms with respect to *RCNT* operators.

Proof:
We need to prove the following cases:

1. $[\![t_1]\!]_\ell \simeq_{rbr} [\![t_2]\!]_\ell$ implies $[\![\alpha.t_1]\!]_\ell \simeq_{rbr} [\![\alpha.t_2]\!]_\ell$;

2. $[\![t_1]\!]_\ell \simeq_{rbr} [\![t_2]\!]_\ell$ and $[\![t_1']\!]_\ell \simeq_{rbr} [\![t_2']\!]_\ell$ implies $[\![t_1 + t_1']\!]_\ell \simeq_{rbr} [\![t_2 + t_2']\!]_\ell$;

3. $[\![t_1]\!]_\ell \simeq_{rbr} [\![t_2]\!]_\ell$ and $[\![t_1']\!]_\ell \simeq_{rbr} [\![t_2']\!]_\ell$ implies $[\![sense(\ell', t_1, t_1')]\!]_\ell \simeq_{rbr} [\![sense(\ell', t_2, t_2')]\!]_\ell$;

4. $[\![t_1]\!]_\ell \simeq_{rbr} [\![t_2]\!]_\ell$ implies $\ell : t : t_1 \simeq_{rbr} \ell : t : t_2$ for any arbitrary term t;
5. $t_1 \simeq_{rbr} t_2$ implies $(\mathcal{C}, \eta).t_1 \simeq_{rbr} (\mathcal{C}, \eta).t_2$;
6. $t_1 \simeq_{rbr} t_2$ and $t'_1 \simeq_{rbr} t'_2$ implies $t_1 + t'_1 \simeq_{rbr} t_2 + t'_2$;
7. $t_1 \simeq_{rbr} t_2$ implies $(\nu\ell)t_1 \simeq_{rbr} (\nu\ell)t_2$;
8. $t_1 \simeq_{rbr} t_2$ and $t'_1 \simeq_{rbr} t'_2$ implies $t_1 \parallel t'_1 \simeq_{rbr} t_2 \parallel t'_2$;
9. $t_1 \simeq_{rbr} t_2$ and $t'_1 \simeq_{rbr} t'_2$ implies $t_1 \between t'_1 \simeq_{rbr} t_2 \between t'_2$;
10. $t_1 \simeq_{rbr} t_2$ and $t'_1 \simeq_{rbr} t'_2$ implies $t_1 \mid t'_1 \simeq_{rbr} t_2 \mid t'_2$;
11. $t_1 \simeq_{rbr} t_2$ implies $\partial_M(t_1) \simeq_{rbr} \partial_M(t_2)$;
12. $t_1 \simeq_{rbr} t_2$ implies $\tau_M(t_1) \simeq_{rbr} \tau_M(t_2)$;
13. $t_1 \simeq_{rbr} t_2$ implies $\mathcal{C} \rhd t_1 \simeq_{rbr} \mathcal{C} \rhd t_2$.

Clearly, if $t_1 \simeq_{rbr} t_2$ then $t_1 \simeq_{br} t_2$ is witnessed by the following branching reliable computed network bisimulation relation:

$$\begin{aligned}\mathcal{R}' = \{\mathcal{R} \mid\ & t_1 \xrightarrow{(\mathcal{C},\eta)} t'_1 \Rightarrow \exists t'_2 \cdot t_2 \xrightarrow{\langle(\mathcal{C},\eta)\rangle} t'_2 \wedge t'_1 \simeq_{br} t'_2 \text{ is witnessed by } \mathcal{R}\}\\ \cup \{\mathcal{R} \mid\ & t_2 \xrightarrow{(\mathcal{C},\eta)} t'_2 \Rightarrow \exists t'_1 \cdot t_1 \xrightarrow{\langle(\mathcal{C},\eta)\rangle} t'_1 \wedge t'_1 \simeq_{br} t'_2 \text{ is witnessed by } \mathcal{R}\}\\ \cup \{&(t_1, t_2)\}.\end{aligned}$$

We prove the cases 1, 2, 4, 7, 10, 11, and 13 since the proof of the cases 3 and 6 are similar to the case 2, the case 5 is similar to the case 1, the cases 8 and 9 are similar to the case 10, and the case 12 is similar to the case 11.

Case 1. The first transitions of $[\![\alpha.t_1]\!]_\ell$ and $[\![\alpha.t_2]\!]_\ell$ are the same with application of the rule Snd (if α is a send action), Rcv_1 (if α is a receive action), or $Rcv_{2,3}$ (for receiving $(\mathcal{C}, nrcv(\mathfrak{m}))$which are not derivable from Rcv_1), and by assumption $[\![t_1]\!]_\ell \simeq_{rbr} [\![t_2]\!]_\ell$ implies $[\![t_1]\!]_\ell \simeq_{br} [\![t_2]\!]_\ell$. Thus the transfer conditions of Definition 5.5 hold.

Case 2. Every transition $[\![t_1 + t'_1]\!]_\ell \xrightarrow{(\mathcal{C},\eta)} t$ owes to $[\![t_1]\!]_\ell \xrightarrow{(\mathcal{C},\eta)} t$ or $[\![t'_1]\!]_\ell \xrightarrow{(\mathcal{C},\eta)} t$ by application of $Choice$, or is implied by application of Rcv_3, i.e., $[\![t_1 + t'_1]\!]_\ell \xrightarrow{(\mathcal{C},nrcv(\mathfrak{m}))} [\![t_1 + t'_1]\!]_\ell$ iff there exists no $t_1 + t'_1 \xrightarrow{(\mathcal{C}',rcv(\mathfrak{m}))} t^*$ for some t^* such that $\mathcal{C}' \preccurlyeq \mathcal{C}$. We assume that $\mathcal{C}$ is the greatest network constraint derived by Rcv_3 as the other can be derived by application of Exe_2. For the former case, $[\![t_1]\!]_\ell \simeq_{rbr} [\![t_2]\!]_\ell$ and $[\![t'_1]\!]_\ell \simeq_{rbr} [\![t'_2]\!]_\ell$ imply there is t' such that $[\![t_2]\!]_\ell \xrightarrow{\langle(\mathcal{C},\eta)\rangle} t'$ or $[\![t'_2]\!]_\ell \xrightarrow{\langle(\mathcal{C},\eta)\rangle} t'$ and $t \simeq_{br} t'$. Thus $[\![t_2 + t'_2]\!]_\ell \xrightarrow{\langle(\mathcal{C},\eta)\rangle} t'$ with $t \simeq_{br} t'$. For the latter case by the rule $Choice$, there exists no $t_1 \xrightarrow{(\mathcal{C}',rcv(\mathfrak{m}))} t^*$ and $t'_1 \xrightarrow{(\mathcal{C}',rcv(\mathfrak{m}))} t^*$ for some t^* such that $\mathcal{C}' \preccurlyeq \mathcal{C}$. Thus by the rule Rcv_3, $[\![t_1]\!]_\ell \xrightarrow{(\mathcal{C},nrcv(\mathfrak{m}))} [\![t_1]\!]_\ell$ and $[\![t'_1]\!]_\ell \xrightarrow{(\mathcal{C},nrcv(\mathfrak{m}))} [\![t'_1]\!]_\ell$. We remark that transitions derived by application of Rcv_3 are those that cannot be derived from $Rcv_{1,2}$. The greatest value of the network

constraints of such transitions have no pair in the form of $? \rightsquigarrow \ell$ or $? \not\rightsquigarrow \ell$. This implies that such transitions can not be mimicked by application of $Rcv_{1,2}$ (since they will add constraints of the form $? \rightsquigarrow \ell$ or $? \not\rightsquigarrow \ell$) . Therefore, $[\![t_1]\!]_\ell \simeq_{rbr} [\![t_2]\!]_\ell$ and $[\![t'_1]\!]_\ell \simeq_{rbr} [\![t'_2]\!]_\ell$ imply that $[\![t_2]\!]_\ell \xrightarrow{(\mathcal{C},nrcv(\mathfrak{m}))} [\![t_2]\!]_\ell$ and $[\![t'_2]\!]_\ell \xrightarrow{(\mathcal{C},nrcv(\mathfrak{m}))} [\![t'_2]\!]_\ell$ which can be only derived by application of the rule Rcv_3. Therefore, there exists no $t_2 \xrightarrow{(\mathcal{C}',rcv(\mathfrak{m}))} t^*$ and $t'_2 \xrightarrow{(\mathcal{C}',rcv(\mathfrak{m}))} t^*$ for some t^* such that $\mathcal{C}' \preccurlyeq \mathcal{C}$. Hence, $[\![t_2 + t'_2]\!]_\ell \xrightarrow{(\mathcal{C},nrcv(\mathfrak{m}))} [\![t_2 + t'_2]\!]_\ell$.

Case 4 Suppose that $\ell : t : t_1 \xrightarrow{(\mathcal{C}^*,\eta)} t^*$. Two cases can be distinguished:

- It owes to the application of one of the rules $Inter'_{1-3}$ together with some of the rules $Choice'$, Inv', and $Sen'_{1,2}$. Therefore, t_1 has a subterm in the form of $\alpha.t'_1$ where $t^* \equiv [\![t'_1]\!]_\ell$, η is the network action version of α, and $\mathcal{C}^* = \mathcal{C}^*_1 \cup \mathcal{C}^*_2$ such that $\mathcal{C}^*_1$ is the network constraint added by $Inter'_{1-3}$ and $\mathcal{C}^*_2$ is added by the rules $Sen'_{1,2}$ if they are applied. By application of $Prefix$ and $Snd/Rcv_{1,2}$ together with the rules $Choice$, Inv, and $Sen_{1,2}$, $[\![t_1]\!]_\ell \xrightarrow{(\mathcal{C}^*,\eta)} [\![t'_1]\!]_\ell$. The assumption $[\![t_1]\!]_\ell \simeq_{rbr} [\![t_2]\!]_\ell$ results that $[\![t_2]\!]_\ell \xrightarrow{\langle(\mathcal{C}^*,\eta)\rangle} [\![t'_2]\!]_\ell$ and $[\![t'_1]\!]_\ell \simeq_{br} [\![t'_2]\!]_\ell$. We remark that a transition of t_1 owing to $Rcv_{1,2}$ cannot be matched to a transition of t_2 generated by Rcv_3. By the rules $Rcv_{1,2}$, two transitions are generated with network constraints of the forms $\{? \rightsquigarrow \ell\} \cup \mathcal{C}$ and $\{? \not\rightsquigarrow \ell\} \cup \mathcal{C}$. Assume that one of these transitions of t_1 is matched to a transition of t_2 generated by Rcv_3 with the network constraint $\mathcal{C}$ (together with Exe_2). Due to our root condition, t_1 must also generate a transition with the network constraint $\mathcal{C}$ by application of Rcv_3 and this is impossible. Thus $[\![t_2]\!]_\ell \xrightarrow{\langle(\mathcal{C}^*,\eta)\rangle} [\![t'_2]\!]_\ell$ implies that t_2 must have a subterm in the form of $\alpha.t'_2$. Therefore with a same discussion, $\ell : t : t_2 \xrightarrow{\langle(\mathcal{C}^*,\eta)\rangle} [\![t'_2]\!]_\ell$ and $[\![t'_1]\!]_\ell \simeq_{br} [\![t'_2]\!]_\ell$.

- It owes to either Sen_3 or Sen_4 as t_1 has a subterm of the form $sense(\ell', t^*_1, t^{**}_1)$. Assume it was derived by Sen_3 (maybe together with $Choice'$, Inv', and $Sen'_{1,2}$), as the other case can be proved with the same argumentation. Thus, the assumptions $\ell : t : t^*_1 \xrightarrow{\not\ nrcv(\mathfrak{m})}$ and $\ell : t : t^{**}_1 \xrightarrow{(\mathcal{C},nrcv(\mathfrak{m}))} [\![t^{**\prime}_1]\!]_\ell$, where $\mathcal{C}^* = \{\ell' \rightsquigarrow \ell\} \cup \mathcal{C}_1$, $\eta^* = nrcv(\mathfrak{m})$ and $t^* = t$ hold. These together imply that $sense(\ell', t^*_1, t^{**}_1) \xrightarrow{\not\ (\{?\rightsquigarrow\ell'\},rcv(\mathfrak{m}))}$ and hence, $t_1 \xrightarrow{\not\ (\{?\rightsquigarrow\ell'\}\cup\mathcal{C}_1[?/\ell],rcv(\mathfrak{m}))}$. Therefore, by application of Rcv_3, $[\![t_1]\!]_\ell \xrightarrow{(\mathcal{C}^*,nrcv(\mathfrak{m}))} [\![t_1]\!]_\ell$, and by application of the rules Sen_2 and Rcv_1 (maybe together with $Choice$, Inv, and $Sen_{1,2}$), $[\![t_1]\!]_\ell \xrightarrow{(\{\ell'\not\rightsquigarrow\ell\}\cup\mathcal{C}\cup\mathcal{C}_1[\ell/?],nrcv(\mathfrak{m}))} [\![t^{**\prime}_1]\!]_\ell$. The assumption $[\![t_1]\!]_\ell \simeq_{rbr} [\![t_2]\!]_\ell$ implies that $[\![t_2]\!]_\ell \xrightarrow{(\{\ell'\not\rightsquigarrow\ell\}\cup\mathcal{C}\cup\mathcal{C}_1[\ell/?],nrcv(\mathfrak{m}))} [\![t^{**\prime}_2]\!]_\ell$, $[\![t^{**\prime}_1]\!]_\ell \simeq_{br} [\![t^{**\prime}_2]\!]_\ell$, and $[\![t_2]\!]_\ell \xrightarrow{(\mathcal{C}^*,nrcv(\mathfrak{m}))} [\![t_2]\!]_\ell$ hold. Due to the root condition, the negative pair $\ell \not\rightsquigarrow \ell'$ can be only derived when t_2 has a subterm of the form $sense(\ell', t^*_2, t^{**}_2)$, where $t^*_2 \xrightarrow{\not\ rcv(\mathfrak{m})}$, $t^{**}_2 \xrightarrow{(\mathcal{C},rcv(\mathfrak{m}))} t^{**\prime}_2$. Consequently $\ell : t : t^*_2 \xrightarrow{\not\ nrcv(\mathfrak{m})}$ and $\ell : t : t^{**}_2 \xrightarrow{(\mathcal{C},nrcv(\mathfrak{m}))} [\![t^{**}_2]\!]_\ell$. By application of Sen_3, $\ell : t : t_2 \xrightarrow{(\mathcal{C}^*,nrcv(\mathfrak{m}))} t$ is achieved.

Case 7. We prove that if $t_1 \simeq_{br} t_2$ then $(\nu\ell)t_1 \simeq_{br} (\nu\ell)t_2$. Let $t_1 \simeq_{br} t_2$ be witnessed by the branching reliable computed network bisimulation relation $\mathcal{R}$. We define $\mathcal{R}' = \{((\nu\ell)t'_1, (\nu\ell)t'_2) | (t'_1, t'_2) \in \mathcal{R}\}$. We prove that $\mathcal{R}'$ is a branching reliable computed network bisimulation relation.

Suppose $(\nu\ell)t'_1 \xrightarrow{(\mathcal{C}',\eta')} (\nu\ell)t''_1$ results from the application of Hid on $t'_1 \xrightarrow{(\mathcal{C},\eta)} t''_1$. Since $(t'_1, t'_2) \in \mathcal{R}$, there are two cases; in the first case η is a τ action and $(t''_1, t'_2) \in \mathcal{R}$, consequently $((\nu\ell)t''_1, (\nu\ell)t'_2) \in \mathcal{R}'$. In the second case there are $s'''_1, \ldots s'''_k$ and $s''_1, \ldots, s''_k$ for some $k > 0$ such that $\forall i \leq k\, (t'_2 \Rightarrow s'''_i \xrightarrow{\langle(\mathcal{C}_i,\eta)\rangle} s''_i$ with $(t'_1, s'''_i), (t''_1, s''_i) \in \mathcal{R})$, and $\langle\mathcal{C}_1\rangle \ldots, \langle\mathcal{C}_k\rangle$ is a partitioning of $\langle\mathcal{C}\rangle$. By application of Hid, $\forall i \leq k\, ((\nu\ell)t'_2 \Rightarrow (\nu\ell)s'''_i$ with $((\nu\ell)t'_1, (\nu\ell)s'''_i) \in \mathcal{R}')$. There are two cases to consider:

- $\langle(\mathcal{C}_i, \eta)\rangle = (\mathcal{C}_i, \eta)$: Consequently $(\nu\ell)s'''_i \xrightarrow{(\mathcal{C}'_i,\eta')} (\nu\ell)s''_i$ where $(\mathcal{C}'_i, \eta') = (\mathcal{C}_i, \eta)[?/\ell]$.
- $\langle(\mathcal{C}_i, \eta)\rangle \neq (\mathcal{C}_i, \eta)$: in this case η is of the form $nsnd(\mathfrak{m}, ?)$, $\eta' = \eta$, and $\mathcal{C}'_i = \mathcal{C}_i[?/\ell]$. If $\langle(\mathcal{C}_i, \eta)\rangle = (\mathcal{C}_i, \eta)[\ell/?]$ then $\langle(\mathcal{C}_i, \eta)\rangle[?/\ell] = (\mathcal{C}'_i, \eta')$ holds, otherwise $\langle(\mathcal{C}_i, \eta)\rangle = (\mathcal{C}_i, \eta)[\ell'/?]$, where $\ell' \neq \ell$, and hence $\langle(\mathcal{C}_i, \eta)\rangle[?/\ell]$ is a counterpart of $(\mathcal{C}'_i, \eta')$. Consequently it holds that $(\nu\ell)s'''_i \xrightarrow{\langle(\mathcal{C}'_i,\eta')\rangle} (\nu\ell)s''_i$.

Owing to the fact that a subset of $\mathcal{C}_1[?/\ell], \ldots, \mathcal{C}_k[?/\ell]$ constitutes a partitioning of $\mathcal{C}[\ell/?]$, and according to the discussion above, there are $s'''_1, \ldots, s'''_j$ and $s''_1, \ldots, s''_j$ for some $j \leq k$ such that $\forall i \leq j$, $(\nu\ell)t'_2 \Rightarrow (\nu\ell)s'''_i \xrightarrow{\langle(\mathcal{C}'_i,\eta')\rangle} (\nu\ell)s''_i$ with $((\nu\ell)t'_1, (\nu\ell)s'''_i), ((\nu\ell)t''_1, (\nu\ell)s''_i) \in \mathcal{R}')$, and $\langle\mathcal{C}'_1\rangle, \ldots, \langle\mathcal{C}'_j\rangle$ is a partitioning of $\langle\mathcal{C}'\rangle$.

Likewise we can prove that $t_1 \simeq_{rbr} t_2$ implies $(\nu\ell)t_1 \simeq_{rbr} (\nu\ell)t_2$. To this aim we examine the root condition in Definition 5.5. Suppose $(\nu\ell)t_1 \xrightarrow{(\mathcal{C}',\eta')} (\nu\ell)t'_1$. With the same argument as above, $(\nu\ell)t_2 \xrightarrow{\langle(\mathcal{C}',\eta')\rangle} (\nu\ell)t'_2$. Since $t'_1 \simeq_{br} t'_2$, we proved that $(\nu\ell)t'_1 \simeq_{br} (\nu\ell)t'_2$. Concluding $(\nu\ell)t_1 \simeq_{rbr} (\nu\ell)t_2$.

Case 10. From the three remaining cases, we focus on the most challenging case, which is the communication merge operator $|$, as the other operators are proved in a similar way. First we prove that if $t_1 \simeq_{br} t_2$, then $t_1 \parallel t \simeq_{br} t_2 \parallel t$.

Let $t_1 \simeq_{br} t_2$ be witnessed by the branching reliable computed network bisimulation relation $\mathcal{R}$. We define $\mathcal{R}' = \{(t'_1 \parallel t', t'_2 \parallel t') \mid (t'_1, t'_2) \in \mathcal{R},\ t' \text{ any computed network term}\}$. We prove that $\mathcal{R}'$ is a branching reliable computed network bisimulation relation. Suppose $t'_1 \parallel t \xrightarrow{(\mathcal{C}^*,\eta)} t^*$. There are several cases to consider:

- Suppose η is of the form $nsnd(\mathfrak{m}, \ell)$. First let it be performed by t'_1, and t participated in the communication. That is, $t'_1 \xrightarrow{(\mathcal{C}_1,nsnd(\mathfrak{m},\ell))} t''_1$ and $t \xrightarrow{(\mathcal{C},nrcv(\mathfrak{m}))} t'$ give rise to the transition $t'_1 \parallel t \xrightarrow{(\mathcal{C}_1\cup C[\ell/?],nsnd(\mathfrak{m},\ell))} t''_1 \parallel t'$. As $(t'_1, t'_2) \in \mathcal{R}$ and $t'_1 \xrightarrow{(\mathcal{C}_1,nsnd(\mathfrak{m},\ell))} t''_1$, there are $s'''_1, \ldots, s'''_k$ and $s''_1, \ldots, s''_k$ for some $k > 0$ such that $\forall i \leq k\, (t'_2 \Rightarrow s'''_i \xrightarrow{(\mathcal{C}_{1_i}[\ell'/\ell],nsnd(\mathfrak{m},\ell'))} s''_i$, where $(\ell =? \vee \ell = \ell')$, with $(t'_1, s'''_i), (t''_1, s''_i) \in \mathcal{R})$, and $\mathcal{C}_{1_1}[\ell'/\ell], \ldots, \mathcal{C}_{1_k}[\ell'/\ell]$ is a partitioning of $\mathcal{C}_1[\ell'/\ell]$. Hence $\forall i \leq k\, (t'_2 \parallel t \Rightarrow s'''_i \parallel t \xrightarrow{((\mathcal{C}_{1_i}[\ell'/\ell]\cup C)[\ell'/?],nsnd(\mathfrak{m},\ell'))} s''_i \parallel t'$ with

$(t_1' \parallel t, s_i''' \parallel t), (t_1'' \parallel t', s_i'' \parallel t') \in \mathcal{R}'$), and $(\mathcal{C}_{1_1}[\ell'/\ell] \cup \mathcal{C})[\ell'/?], \ldots, (\mathcal{C}_{1_k}[\ell'/\ell] \cup \mathcal{C})[\ell'/?]$ is a partitioning of $(\mathcal{C}_1[\ell'/\ell] \cup \mathcal{C})[\ell'/?]$.

Now suppose that the send action was performed by t, and t_1' participated in the communication. That is, $t_1' \xrightarrow{(\mathcal{C}_1, nrcv(\mathfrak{m}))} t_1''$ and $t \xrightarrow{(\mathcal{C}, nsnd(\mathfrak{m},\ell))} t'$ give rise to the transition $t_1' \parallel t \xrightarrow{(\mathcal{C}_1 \cup \mathcal{C}[\ell/?], nsnd(\mathfrak{m},\ell))} t_1'' \parallel t'$. Since $(t_1', t_2') \in \mathcal{R}$ and $t_1' \xrightarrow{(\mathcal{C}_1, nrcv(\mathfrak{m}))} t_1''$, there are $s_1''', \ldots, s_k'''$ and $s_1'', \ldots, s_k''$ for some $k > 0$ such that $\forall i \leq k\,(t_2' \Rightarrow s_i''' \xrightarrow{(\mathcal{C}_{1_i}, nrcv(\mathfrak{m}))} s_i''$ with $(t_1', s_i'''), (t_1'', s_i'') \in \mathcal{R})$, and $\mathcal{C}_{1_1}, \ldots, \mathcal{C}_{1_k}$ is a partitioning of $\mathcal{C}_1$. Therefore, $\forall i \leq k\,(t_2' \parallel t \Rightarrow s_i''' \parallel t \xrightarrow{(\mathcal{C}_{1_i} \cup C[\ell/?], nsnd(\mathfrak{m},\ell))} s_i'' \parallel t'$, and $(t_1' \parallel t, s_i''' \parallel t), (t_1'' \parallel t', s_i'' \parallel t') \in \mathcal{R}')$ and $\mathcal{C}_{1_1} \cup \mathcal{C}[\ell/?], \ldots, \mathcal{C}_{1_k} \cup \mathcal{C}[\ell/?]$ constitute a partitioning of $\mathcal{C}_1 \cup \mathcal{C}[\ell/?]$.

- The case where η is a receive action is proved in a similar way to the previous case.

- Suppose η is a τ action. Assume it originates from t_1 by application of Par. Thus $t_1' \xrightarrow{(\mathcal{C},\tau)} t_1''$ and $(t_1', t_2') \in \mathcal{R}$ implies: either $(t_1'', t_2') \in \mathcal{R}$ and consequently $(t_1'' \parallel t, t_2' \parallel t) \in \mathcal{R}'$, or there are $s_1''', \ldots, s_k'''$ and $s_1'', \ldots, s_k''$ for some $k > 0$ such that $\forall i \leq k\,(t_2' \Rightarrow s_i''' \xrightarrow{(\mathcal{C}_i,\tau)} s_i''$ with $(t_1', s_i'''), (t_1'', s_i'') \in \mathcal{R})$, and $\mathcal{C}_1, \ldots, \mathcal{C}_k$ constitute a partitioning of $\mathcal{C}$. Therefore, $\forall i \leq k\,(t_2' \parallel t \Rightarrow s_i''' \parallel t \xrightarrow{(C_i,\tau)} s_i'' \parallel t'$, and $(t_1' \parallel t, s_i''' \parallel t), (t_1'' \parallel t', s_i'' \parallel t') \in \mathcal{R}')$. The case when $t \xrightarrow{(\mathcal{C},\tau)} t'$ implies $t_1' \parallel t \xrightarrow{(\mathcal{C},\tau)} t_1' \parallel t'$ by application of Par is straightforward.

- The case when η is an internal action is easy to prove (similar to the second case of the previous case).

Likewise we can prove that $t_1 \simeq_{rbr} t_2$ implies $t \parallel t_1 \simeq_{rbr} t \parallel t_2$.

Now let $t_1 \simeq_{rbr} t_2$. To prove $t_1 \mid t \simeq_{rbr} t_2 \mid t$, we examine the root condition from Definition 5.5. Suppose $t_1 \mid t \xrightarrow{(\mathcal{C}^*, nsnd(\mathfrak{m},\ell))} t^*$. There are two cases to consider:

- This send action was performed by t_1 at node ℓ, and t participated in the communication. That is, $t_1 \xrightarrow{(\mathcal{C}_1, nsnd(\mathfrak{m},\ell))} t_1'$ and $t \xrightarrow{(\mathcal{C}, nrcv(\mathfrak{m}))} t'$, so that $t_1 \mid t \xrightarrow{(\mathcal{C}_1 \cup \mathcal{C}[\ell/?], nsnd(\mathfrak{m},\ell))} t_1' \parallel t'$. Since $t_1 \simeq_{rbr} t_2$, there is a t_2' such that $t_2 \xrightarrow{(\mathcal{C}_1, nsnd(\mathfrak{m},\ell'))} t_2'$ with $(\ell =? \vee \ell = \ell')$ and $t_1' \simeq_{br} t_2'$. Then $t_2 \mid t \xrightarrow{(\mathcal{C}_1 \cup \mathcal{C}[\ell'/?], nsnd(\mathfrak{m},\ell))} t_2' \parallel t'$. Since $t_1' \simeq_{br} t_2'$, we proved that $t_1' \parallel t' \simeq_{br} t_2' \parallel t'$.

- The send action was performed by t at node ℓ, and t_1 participated in the communication. That is, $t_1 \xrightarrow{(\mathcal{C}_1, nrcv(\mathfrak{m}))} t_1'$ and $t \xrightarrow{(\mathcal{C}, nsnd(\mathfrak{m},\ell))} t'$, so that $t_1 \mid t \xrightarrow{(\mathcal{C}_1 \cup \mathcal{C}[\ell/?], nsnd(\mathfrak{m},\ell))} t_1' \parallel t'$. Since $t_1 \simeq_{rbr} t_2$, there is a t_2' such that $t_2 \xrightarrow{(\mathcal{C}_1, nrcv(\mathfrak{m}))} t_2'$ with $t_1' \simeq_{br} t_2'$. Then $t_2 \mid t \xrightarrow{(\mathcal{C}_1 \cup \mathcal{C}[\ell/?], nsnd(\mathfrak{m},\ell))} t_2' \parallel t'$. Since $t_1' \simeq_{br} t_2'$, we have $t_1' \parallel t' \simeq_{br} t_2' \parallel t'$.

Finally, the case where $t_1 \mid t \xrightarrow{(\mathcal{C}^*, nrcv(\mathfrak{m}))} t^*$ can be easily dealt with. This receive action was performed by both t_1 and t.

Concluding, $t_1 \mid t \simeq_{rbr} t_2 \mid t$. Likewise it can be argued that $t \mid t_1 \simeq_{rbr} t \mid t_2$.

Case 11. We prove that if $t_1 \simeq_{br} t_2$, then $\partial_M(t_1) \simeq_{br} \partial_M(t_2)$. Let $t_1 \simeq_{br} t_2$ be witnessed by the branching reliable computed network bisimulation relation $\mathcal{R}$. We define $\mathcal{R}' = \{(\partial_M(t'_1), \partial_M(t'_2)) \mid (t'_1, t'_2) \in \mathcal{R}\}$. We prove that $\mathcal{R}'$ is a branching reliable computed network bisimulation relation. Suppose that $\partial_M(t'_1) \xrightarrow{(\mathcal{C},\eta)} \partial_M(t''_1)$ results from the application of $Encap$ on $t'_1 \xrightarrow{(\mathcal{C},\eta)} t''_1$ such that $\eta \neq nrcv(\mathfrak{m}) \vee isType_m(\mathfrak{m}) = F$. Since $(t'_1, t'_2) \in \mathcal{R}$, two cases can be considered: either η is a τ action and $(t''_1, t'_2) \in \mathcal{R}$, or there are $s'''_1, \ldots, s'''_k$ and $s''_1, \ldots, s''_k$ for some $k > 0$ such that $\forall i \leq k\,(t'_2 \Rightarrow s'''_i \xrightarrow{\langle(\mathcal{C}_i,\eta)\rangle} s''_i$ with $(t'_1, s'''_i), (t''_1, s''_i) \in \mathcal{R})$ and $\langle\mathcal{C}_1\rangle, \ldots, \langle\mathcal{C}_k\rangle$ is a partitioning of the network constraint $\langle\mathcal{C}\rangle$. In the former case, $(\partial_M(t''_1), \partial_M(t'_2)) \in \mathcal{R}'$. In the latter case, by application of Par and $Encap$, $\forall i \leq k\,(\partial_M(t'_2) \Rightarrow \partial_M(s'''_i) \xrightarrow{\langle(\mathcal{C}_i,\eta)\rangle} \partial_M(t''_2)$ with $(\partial_M(t'_1), \partial_M(s'''_i)), (\partial_M(t''_1), \partial_M(s''_i)) \in \mathcal{R}')$.

Likewise we can prove that $t_1 \simeq_{rbr} t_2$ implies $\partial_M(t_1) \simeq_{rbr} \partial_M(t_2)$. To this aim we examine the root condition in Definition 5.5. Suppose $\partial_M(t_1) \xrightarrow{(\mathcal{C},\eta)} \partial_M(t'_1)$. With the same argument as above, $\partial_M(t_2) \xrightarrow{\langle(\mathcal{C},\eta)\rangle} \partial_M(t'_2)$. Since $t'_1 \simeq_{br} t'_2$, we proved that $\partial_M(t'_1) \simeq_{br} \partial_M(t'_2)$. Concluding that $\partial_M(t_1) \simeq_{rbr} \partial_M(t_2)$.

Case 13 Suppose that $\mathcal{C} \rhd t_1 \xrightarrow{(\mathcal{C}'\cup\mathcal{C},\eta)} t'_1$ by application of TR, since $t_1 \xrightarrow{(\mathcal{C}',\eta)} t'_1$. By assumption $t_1 \simeq_{rbr} t_2$ implies that $t_2 \xrightarrow{(\mathcal{C}',\eta)} t'_2$ and $t'_1 \simeq_{br} t'_2$. Therefore, by application of the rule TR, $\mathcal{C} \rhd t_2 \xrightarrow{(\mathcal{C}'\cup\mathcal{C},\eta)} t'_2$, and $t'_1 \simeq_{br} t'_2$ concludes that $\mathcal{C} \rhd t_1 \simeq_{rbr} \mathcal{C} \rhd t_2$. □

C. Soundness of *RCNT* axiomatization

As two rooted branching computed network bisimilar terms are also rooted branching reliable computed network bisimilar, the soundness of axioms which are in common with the lossy setting are established [21]. Thus, to prove Theorem 6.1, it suffices to prove the soundness of each new axiom in comparison with the lossy setting, i.e., Dep_{0-7}, $TRes_{1-5}$, $LM'_{1,2}$, and T_1, modulo rooted branching reliable computed network bisimilarity.

We focus on the soundness of Dep_0 and T_1, as the soundness of the remaining axioms can be argued in a similar fashion. To prove Dep_0, we show that both sides of the axiom satisfy the transfer conditions of Definition 5.5. In following cases, for the sake of brevity, we write X for $rec\mathfrak{Q} \cdot \sum_{\mathfrak{m}' \notin Message(t,\emptyset)}(\{\}, nrcv(\mathfrak{m}')).\mathfrak{Q} + \ell : \mathfrak{Q} : t$. Assume that $[\![t]\!]_\ell \xrightarrow{(\mathcal{C}^*,\eta)} [\![t']\!]_\ell$. Two cases can be distinguished:

1. It owes to the application $Prefix$ and $Snd/Rcv_{1,2}$ together with some of the rules $Choice$, Inv, and $Sen_{1,2}$. Therefore t has a subterm of the form of $\alpha.t'$ where η is the network action version of α and $\mathcal{C}^* \equiv \mathcal{C}^*_1 \cup \mathcal{C}^*_2$ such that $\mathcal{C}^*_1$ is derived by $Snd/Rcv_{1,2}$ and $\mathcal{C}^*_2$ is derived by $Sen_{1,2}$ if they are applied. By application of $Inter'_1/Inter'_{2,3}$ together with some of the rules $Choice'$, Inv', and $Sen'_{1,2}$, $\ell : X : t \xrightarrow{(\mathcal{C}^*,\eta)} [\![t']\!]_\ell$. Then, by application of Rec and $Choice$, $X \xrightarrow{(\mathcal{C}^*,\eta)} [\![t']\!]_\ell$.

2. It owes to the application of Rcv_3 since $t \xrightarrow{(\mathcal{C},rcv(\mathfrak{m}))}\!\!\!\!\!\!/\;$, where $\mathcal{C}^* = \mathcal{C}[\ell/?]$, and there exists no t' such that $t \xrightarrow{(\mathcal{C}',\ rcv(\mathfrak{m}))}\!\!\!\!\!\!/\; t' \wedge \mathcal{C}' \preccurlyeq \mathcal{C}$. Two cases can be distinguished:
   - Assume that $m \notin Message(t, \emptyset)$. Thus, by application of Rec, $Choice$ and $Prefix$, $X \xrightarrow{(\mathcal{C},nrcv(\mathfrak{m}))} X$, where $\mathcal{C} = \{\}$.
   - Assume that $m \in Message(t, \emptyset)$ and consequently $t \xrightarrow{(\mathcal{C}'',rcv(\mathfrak{m}))} t'$ for some t'. This only happens when t has a subterm of the form of $sense(\ell', t_1, t_2)$ for some t_1 and t_2. Assume that $t_1 \xrightarrow{(\mathcal{C}'''[?/\ell],rcv(\mathfrak{m}))} t_1'$, where $\{? \rightsquigarrow \ell'\} \cup \mathcal{C}'''[?/\ell] \cup \mathcal{C}_1^*[?/\ell] = \mathcal{C}''$ and $t_2 \xrightarrow{(\{\},\ rcv(\mathfrak{m}))}\!\!\!\!\!\!/\;$. Thus, $t \xrightarrow{(\{?\rightsquigarrow\ell'\}\cup\mathcal{C}_1^*[?/\ell],rcv(\mathfrak{m}))}\!\!\!\!\!\!/\;$ and $\mathcal{C} \equiv \{? \rightsquigarrow \ell'\} \cup \mathcal{C}_1^*[?/\ell]$. Therefore, $\ell : X : t_1 \xrightarrow{(\mathcal{C}'''\cup\{\ell\rightsquigarrow\ell'\},nrcv(\mathfrak{m}))} [\![t_1']\!]_\ell$ and $\ell : X : t_2 \xrightarrow{nrcv(\mathfrak{m})}\!\!\!\!\!\!/\;$, and by application of Sen_4 (and maybe together with $Choice'$, Inv', and $Sen'_{1,2}$), $\ell : X : t \xrightarrow{(\{\ell\not\rightsquigarrow\ell'\}\cup\mathcal{C}_1^*,nrcv(\mathfrak{m}))} X$. The case $t_2 \xrightarrow{(\mathcal{C}'''[?/\ell],rcv(\mathfrak{m}))} t_2'$, where $\{? \rightsquigarrow \ell'\} \cup \mathcal{C}'''[?/\ell] \cup \mathcal{C}_1^*[?/\ell] = \mathcal{C}''$, and $t_1 \xrightarrow{(\{\},\ rcv(\mathfrak{m}))}\!\!\!\!\!\!/\;$ hold, is proved with a similar discussion with application of Sen_3.

We focus on the soundness of T_1. The only transition that $(\mathcal{C}', \eta).((\mathcal{C}_1, \eta).t + (\mathcal{C}_2, \eta).t + t')$ and $(\mathcal{C}', \eta).((\mathcal{C}, \eta).t + t')$ in T_1 can do is $\xrightarrow{(\mathcal{C}',\eta)}$ and the resulting terms $(\mathcal{C}_1, \eta).t + (\mathcal{C}_2, \eta).t + t'$ and $(\mathcal{C}, \eta).t + t'$ are branching reliable computed network bisimilar, witnessed by the relation $\mathcal{R}$ constructed as follows:

$$\mathcal{R} = \{((\mathcal{C}_1, \eta).t + (\mathcal{C}_2, \eta).t + t', (\mathcal{C}, \eta).t + t'), (t, t) \mid t \in RCNT\}.$$

The pair $((\mathcal{C}_1, \eta).t + (\mathcal{C}_2, \eta).t + t', (\mathcal{C}, \eta).t + t')$ satisfies the transfer conditions of Definition 5.3. Because every initial transition that $(\mathcal{C}_1, \eta).t + (\mathcal{C}_2, \eta).t + t'$ can perform owing to t', $(\mathcal{C}, \eta).t + t'$ can perform too. If $(\mathcal{C}_1, \eta).t + (\mathcal{C}_2, \eta).t + t'$ can perform a $(\mathcal{C}_1, \eta)$ or $(\mathcal{C}_2, \eta)$-transition, $(\mathcal{C}, \eta).t + t'$ can also perform it by application of Exe. Vice versa, if $(\mathcal{C}, \eta).t + t'$ can perform a $(\mathcal{C}, \eta)$-transition, then as $\mathcal{C}_1$ and $\mathcal{C}_2$ form a partitioning of $\mathcal{C}$, $(\mathcal{C}_1, \eta).t + (\mathcal{C}_2, \eta).t + t'$ can perform a corresponding $(\mathcal{C}_1, \eta)$- or $(\mathcal{C}_2, \eta)$-transition.

D. Completeness of *RCNT* axiomatization

To define *RCNT* terms with a finite-state behavior, we borrow the syntactical restriction of [21] on recursive terms $rec\mathfrak{A} \cdot t$, following the approach of [1]. We consider so-called *finite-state Reliable Computed Network Theory* (*RCNT*$_f$), obtained by restricting recursive terms $rec\mathfrak{A} \cdot t$ to those that of which the bound network names do not occur in the scope of parallel, communication merge, left merge, hide, encapsulation and abstraction operators in t.

We follow the corresponding proof of [21] to prove Theorem 6.2 by performing the following steps:

1. first we show that each *RCNT*$_f$ term can be turned into a *normal form* consisting of only $0, (\mathcal{C}, \eta).t', t' + t''$ and $rec\mathfrak{A} \cdot t'$, where $\mathfrak{A}$ is guarded in t';

2. next we define *recursive network specification*s and prove that each guarded recursive network specification has a unique solution;

3. finally we show that our axiomatization is ground-complete for normal forms, by showing that equivalent normal forms are solutions for the same guarded recursive network specification.

Completeness of our axiomatization for all $RCNT_f$ terms results from the steps 1 and 3. We only discuss the first step, as others are exactly the same as in the lossy setting.

Proposition D.1. Each closed term t of $RCNT_f$ whose network names do not occur in the scope of one of the operators $\|, \between, |, (\nu\ell), \tau_M$ or ∂_M for some $\ell \in Loc$ and $M \subseteq Msg$, can be turned into a normal form.

We prove this by structural induction over the syntax of terms t (possibly open). The base cases of induction for $t \equiv 0$ or $t \equiv \mathfrak{A}$ are trivial because they are in normal form already. The inductive cases of the induction are the following ones:

- if $t \equiv [\![0]\!]_\ell$, then by application of $Dep_{0,4}$ and Ch_1 we have $t = rec\mathfrak{Q}\cdot\sum_{\mathfrak{m}' \notin Msg}(\{\}, nrcv(\mathfrak{m}')).\mathfrak{Q}$, which is in normal form.

- if $t \equiv [\![\alpha.t']\!]_\ell$ or $t \equiv [\![t' + t'']\!]_\ell$ or $[\![sense(\ell', t', t'')]\!]_\ell$ or $[\![\mathfrak{A}]\!]_\ell$, then t can be turned into a normal form by application of axioms $Dep_{0-5,6,7}$ and induction over $[\![t']\!]_\ell$ and $[\![t'']\!]_\ell$.

- if $t \equiv (\mathcal{C}, \eta).t'$ or $t \equiv t' + t''$, then t can be turned into normal form by induction over t' and t''.

- the other cases can be treated in the same way as in [21].

E. Proofs of section 7.2

We first prove Theorem 7.2 which indicates that the refinement relation is a preorder relation and has the precongruence property, and then we discuss the proof of Proposition 7.3.

E.1. Proof of theorem 7.2

We first show that the refinement relation is a preorder relation and then discuss its precongruence property. To prove that refinement is a preorder, we must show that it is reflexive and transitive. As it is trivial that Definition 7.1 is reflexive, we focus on its transitivity property.

Regrading the well-formedness conditions imposed on *RCNT* terms, the transitivity property of our refinement relation, i.e., $t_1 \sqsubseteq t_2$ and $t_2 \sqsubseteq s$ implies that $t_1 \sqsubseteq s$, can be only proved when t_1 and t_2 have no prefixed-actions with a multi-hop network constraint. For such terms, Definition 7.1 enforces they mimic the behavior of each other by the second case of first and second transfer condition (as long as accumulated network constraints are well-formed). In other words, for reliable computed network terms with no prefixed-actions with multi-hop network constraints, a relation which is strong bisimulation of [44] is also a refinement.

Lemma E.1. (Transitive property)
$t_1 \sqsubseteq t_2$ and $t_2 \sqsubseteq s$ implies that $t_1 \sqsubseteq s$.

Proof:
Assume sets of refinement relations $\mathcal{R}^1_{\mathcal{C}}$ and $\mathcal{R}^2_{\mathcal{C}}$ witnessing $t_1 \sqsubseteq t_2$ and $t_2 \sqsubseteq s$, respectively. We construct a set of refinement relations $\mathcal{R}'_{\mathcal{C}} = \{(t'_1, s') \mid (t'_2, s') \in \mathcal{R}^2_{\mathcal{C}} \wedge t'_1 \mathcal{R}^1_{\mathcal{C}} t_2\}$ for any well-formed network constraint $\mathcal{C}$. We show that $t'_1 \mathcal{R}'_{\mathcal{C}} s'$ satisfies the transfer conditions of Definition 7.1.

Assume $t'_1 \xrightarrow{(\mathcal{C}',\eta)} t''_1$ where $\mathcal{C} \cup \mathcal{C}' \in \mathbb{C}^v(Loc)$. By assumption $t'_1 \mathcal{R}^1_{\mathcal{C}} t'_2$ implies that $t'_2 \xrightarrow{(\mathcal{C}',\eta)} t''_2$ such that $t''_1 \mathcal{R}^1_{\mathcal{C}\cup\mathcal{C}'} t''_2$. By the assumption $t'_2 \mathcal{R}^2_{\mathcal{C}} s'$, there are three cases to consider:

- $\eta = \tau$ and $t''_2 \; \mathcal{R}^2_{\mathcal{C}\cup\mathcal{C}'} \; s'$. Thus by construction, $t''_1 \; \mathcal{R}'_{\mathcal{C}\cup\mathcal{C}'} \; s'$.
- There is an s'' such that $s' \xrightarrow{(\mathcal{C},\eta)} s''$, and $t''_2 \; \mathcal{R}^2_{\mathcal{C}\cup\mathcal{C}'} \; s''$. Thus by construction, $t''_1 \; \mathcal{R}'_{\mathcal{C}\cup\mathcal{C}'} \; s'$.
- $\eta = \iota$ for some $\iota \in IAct \cup \{\tau\}$ and there is an s'' such that $s' \xrightarrow{(\mathcal{M},\iota)} s''$ with $\mathcal{C} \cup \mathcal{C}' \models \mathcal{M}$ and $t''_2 \; \mathcal{R}^2_{\mathcal{C}\cup\mathcal{C}'} \; s''$. Thus by construction, $t''_1 \; \mathcal{R}'_{\mathcal{C}\cup\mathcal{C}'} \; s''$.

Assume $s' \xrightarrow{(\mathcal{C}',\eta)} s''$. The assumption $t'_2 \; \mathcal{R}^2_{\mathcal{C}\cup\mathcal{C}'} \; s'$ implies that there is a t''_2 such that $t'_2 \xrightarrow{(\mathcal{C}',\eta)} t''_2$ with $t''_2 \; \mathcal{R}^2_{\mathcal{C}\cup\mathcal{C}'} \; s''$. By assumption $t'_1 \mathcal{R}^1_{\mathcal{C}} t'_2$ implies that $t'_1 \xrightarrow{(\mathcal{C}',\eta)} t''_1$ such that $t''_1 \mathcal{R}^1_{\mathcal{C}\cup\mathcal{C}'} t''_2$, and consequently $t''_1 \; \mathcal{R}'_{\mathcal{C}\cup\mathcal{C}'} \; s''$.

Assume $s' \xrightarrow{(\mathcal{M},\iota)} s''$. . The assumption $t'_2 \; \mathcal{R}^2_{\mathcal{C}\cup\mathcal{C}'} \; s'$ implies that there are t'''_2 and t''_2 such that $t'_2 \stackrel{\mathcal{C}'}{\Longrightarrow} t'''_2 \xrightarrow{(\mathcal{C}'',\iota)} t''_2$ with $t'''_2 \; \mathcal{R}^2_{\mathcal{C}\cup\mathcal{C}'} \; s'$ and $t''_2 \; \mathcal{R}^2_{\mathcal{C}\cup\mathcal{C}'\cup\mathcal{C}''} \; s''$ where $\mathcal{C} \cup \mathcal{C}' \cup \mathcal{C}'' \models \mathcal{M}$. As every transition of t'_2 is mimicked by t'_1, there are t'''_1 and t''_1 such that $t'_1 \stackrel{\mathcal{C}'}{\Longrightarrow} t'''_1 \xrightarrow{(\mathcal{C}'',\iota)} t''_1$ with $t'''_1 \; \mathcal{R}^1_{\mathcal{C}\cup\mathcal{C}'} \; t'''_2$ and $t''_1 \; \mathcal{R}^1_{\mathcal{C}\cup\mathcal{C}'\cup\mathcal{C}''} \; t''_2$. Concluding, there are t'''_1 and t''_1 such that $t'_1 \stackrel{\mathcal{C}'}{\Longrightarrow} t'''_1 \xrightarrow{(\mathcal{C}'',\iota)} t''_1$ with $t'''_1 \; \mathcal{R}'_{\mathcal{C}\cup\mathcal{C}'} \; s'$ and $t''_1 \; \mathcal{R}'_{\mathcal{C}\cup\mathcal{C}'\cup\mathcal{C}''} \; s''$ where $\mathcal{C} \cup \mathcal{C}' \cup \mathcal{C}'' \models \mathcal{M}$. □

Theorem E.2. Refinement is a precongruence for terms with respect to the *RCNT* operators.

Proof:
Assume that $t_1 \sqsubseteq s_1$ and $t_2 \sqsubseteq s_2$. We first show that $t_1 + t_2 \sqsubseteq s_1 + s_2$. There are sets of refinement relations $\mathcal{R}^1_{\mathcal{C}}$ and $\mathcal{R}^2_{\mathcal{C}}$ witnessing $t_1 \sqsubseteq s_1$ and $t_2 \sqsubseteq s_2$, respectively. We construct a set of refinement relations $\mathcal{R}_{\mathcal{C}} = \mathcal{R}^1_{\mathcal{C}} \cup \mathcal{R}^2_{\mathcal{C}} \cup \{(t'_1, s_1 + s_2) \mid t'_1 \; \mathcal{R}^1_{\mathcal{C}} \; s_1\} \cup \{(t'_2, s_1 + s_2) \mid t'_2 \; \mathcal{R}^2_{\mathcal{C}} \; s_2\}$ for any well-formed network constraint $\mathcal{C}$. We show that $\mathcal{R}_{\{\,\}} = \{(t_1 + t_2, s_1 + s_2)\} \cup \mathcal{R}^1_{\{\,\}} \cup \mathcal{R}^2_{\{\,\}}$ satisfies the transfer conditions of Definition 7.1.

Assume $t_1 + t_2 \xrightarrow{(\mathcal{C}',\eta)} t'_1$ owing to $t_1 \xrightarrow{(\mathcal{C}',\eta)} t'_1$, where $\mathcal{C} \cup \mathcal{C}' \in \mathbb{C}^v(Loc)$. By the assumption $t_1 \; \mathcal{R}^1_{\{\,\}} \; s_1$, three cases can be considered:

- $\eta = \tau$ and $t'_1 \; \mathcal{R}_{\mathcal{C}\cup\mathcal{C}'} \; s_1$. Thus by construction $t'_1 \; \mathcal{R}_{\mathcal{C}\cup\mathcal{C}'} \; s_1 + s_2$.
- There is an s'_1 such that $s_1 \xrightarrow{(\mathcal{C},\eta)} s'_1$, and $t'_1 \; \mathcal{R}^1_{\mathcal{C}\cup\mathcal{C}'} \; s'_1$. Thus by the rule $Choice$, there is an s'_1 such that $s_1 + s_2 \xrightarrow{(\mathcal{C},\eta)} s'_1$ and by construction $t'_1 \; \mathcal{R}_{\mathcal{C}\cup\mathcal{C}'} \; s'_1$.

- $\eta = \iota$ for some $\iota \in IAct \cup \{\tau\}$ and there is an s_1' such that $s_1 \xrightarrow{(\mathcal{M},\iota)} s_1'$ with $\mathcal{C} \cup \mathcal{C}' \models \mathcal{M}$ and $t_1' \; \mathcal{R}^1_{\mathcal{C}\cup\mathcal{C}'} \; s_1'$. Thus by the rule *Choice*, there is an s_1' such that $s_1 + s_2 \xrightarrow{(\mathcal{M},\iota)} s_1'$ and by construction $\mathcal{C} \cup \mathcal{C}' \models \mathcal{M}$ and $t_1' \; \mathcal{R}_{\mathcal{C}\cup\mathcal{C}'} \; s_1'$.

The same discussion holds if $t_1 + t_2 \xrightarrow{(\mathcal{C}',\eta)} t_2'$ owing to $t_2 \xrightarrow{(\mathcal{C}',\eta)} t_2'$.

Assume $s_1 + s_2 \xrightarrow{(\mathcal{M},\iota)} s_1'$ owing to $s_1 \xrightarrow{(\mathcal{M},\iota)} s_1'$, where $\mathcal{C} \cup \mathcal{C}' \in \mathbb{C}^v(Loc)$. By assumption $t_1 \; \mathcal{R}^1_{\mathcal{C}} \; s_1$ implies there are t_1'' and t_1' such that $t_1 \stackrel{\mathcal{C}'}{\Longrightarrow} t_1'' \xrightarrow{(\mathcal{C}'',\iota)} t_1'$ with $t_1'' \; \mathcal{R}^1_{\mathcal{C}\cup\mathcal{C}'} \; s_1$ and $t_1' \; \mathcal{R}^1_{\mathcal{C}\cup\mathcal{C}'\cup\mathcal{C}''} \; s_1'$ where $\mathcal{C} \cup \mathcal{C}' \cup \mathcal{C}'' \models \mathcal{M}$. Consequently $t_1 + t_2 \stackrel{\mathcal{C}'}{\Longrightarrow} t_1'' \xrightarrow{(\mathcal{C}'',\iota)} t_1'$ with $t_1'' \; \mathcal{R}_{\mathcal{C}\cup\mathcal{C}'} \; s_1 + s_2$ and $t_1' \; \mathcal{R}_{\mathcal{C}\cup\mathcal{C}'\cup\mathcal{C}''} \; s_1'$ where $\mathcal{C} \cup \mathcal{C}' \cup \mathcal{C}'' \models \mathcal{M}$. The same discussion holds when $s_1 + s_2 \xrightarrow{(\mathcal{M},\iota)} s_2'$ owing to $s_2 \xrightarrow{(\mathcal{M},\iota)} s_2'$.

Assume $s_1 + s_2 \xrightarrow{(\mathcal{C},\eta)} s_1'$ owing to $s_1 \xrightarrow{(\mathcal{C},\eta)} s_1'$. By assumption $t_1 \; \mathcal{R}^1_{\mathcal{C}} \; s_1$ implies there is a t_1' such that $t_1 \xrightarrow{(\mathcal{C}',\eta)} t_1'$ with $t_1' \; \mathcal{R}^1_{\mathcal{C}\cup\mathcal{C}'} \; s_1'$. Hence, there is a t_1' such that $t_1 + t_2 \xrightarrow{(\mathcal{C}',\eta)} t_1'$ with $t_1' \; \mathcal{R}_{\mathcal{C}\cup\mathcal{C}'} \; s_1'$.

The above discussions together yield $t_1 + t_2 \sqsubseteq s_1 + s_2$.

If s_1 and s_2 have no prefixed-action with a multi-hop network constraint, then we must show the following cases:

1. $(\mathcal{C}, \eta).t_1 \sqsubseteq (\mathcal{C}, \eta).t_2$;
2. $(\nu\ell).t_1 \sqsubseteq (\nu\ell).t_2$;
3. $t_1 \parallel t_2 \sqsubseteq s_1 \parallel s_2$;
4. $t_1 \between t_2 \sqsubseteq s_1 \between s_2$;
5. $t_1 \mid t_2 \sqsubseteq s_1 \mid s_2$;
6. $\partial_M(t_1) \sqsubseteq \partial_M(t_2)$;
7. $\tau(t_1) \sqsubseteq \tau(t_2)$;
8. $\mathcal{C} \rhd t_1 \sqsubseteq \mathcal{C} \rhd t_2$;

The above cases result from the congruence property of strong bisimilarity. As we discussed earlier, for reliable computed network terms with no prefixed-actions with multi-hop network constraints, a relation which is strong bisimulation of [44] is also a refinement. □

The proof of Theorem 7.2 is an immediate result of Lemma E.1 and Theorem E.2.

E.2. Proof of proposition 7.3

First we show that $(\mathcal{C},\tau).t \sqsubseteq (\mathcal{M},\iota).s \Rightarrow \mathcal{C} \rhd t \sqsubseteq (\mathcal{M},\iota).s \wedge \mathcal{C} \models \mathcal{M}$. The only transition $(\mathcal{C},\tau).t$ can make is $(\mathcal{C},\tau).t \xrightarrow{(\mathcal{C},\tau)} t$. As $\iota \neq \tau$, according to the first case of the first transfer condition of Definition 7.1, $t\ \mathcal{R}_{\mathcal{C}}\ (\mathcal{M},\iota).s$. We construct $\mathcal{R}'_{\{\ \}} = \mathcal{R}_{\mathcal{C}}$ and show that it induces $\mathcal{C} \rhd t \sqsubseteq (\mathcal{M},\iota).s$. This is trivial as any transition $\mathcal{C} \rhd t \xrightarrow{(\mathcal{C}\cup\mathcal{C}',\eta)} t'$ is the result of $t \xrightarrow{(\mathcal{C}',\eta)} t'$. The transition $s \xrightarrow{(\mathcal{M},\iota)} s'$ and the assumption $t\ \mathcal{R}_{\mathcal{C}}\ (\mathcal{M},\iota).s$ imply that here are t'' and t' such that $t \stackrel{\mathcal{C}'}{\Longrightarrow} t'' \xrightarrow{(\mathcal{C}'',\iota)} t'$ with $\mathcal{C}\cup\mathcal{C}'\cup\mathcal{C}'' \models \mathcal{M}$ and $t'\ \mathcal{R}_{\mathcal{C}\cup\mathcal{C}'\cup\mathcal{C}''}\ s'$ where $\mathcal{C}\cup\mathcal{C}'\cup\mathcal{C}'' \in \mathbb{C}^v(Loc)$. Two cases can be discussed:

- $t'' \equiv t$, and $t \stackrel{\{\ \}}{\Longrightarrow} t \xrightarrow{(\mathcal{C}'',\iota)} t'$ with $\mathcal{C}\cup\mathcal{C}'' \models \mathcal{M}$ and $t'\ \mathcal{R}_{\mathcal{C}\cup\mathcal{C}''}\ s'$ where $\mathcal{C}\cup\mathcal{C}'' \in \mathbb{C}^v(Loc)$. Therefore, $\mathcal{C} \rhd t \stackrel{\{\ \}}{\Longrightarrow} \mathcal{C} \rhd t \xrightarrow{(\mathcal{C}''\cup\mathcal{C},\iota)} t'$ with $\mathcal{C}\cup\mathcal{C}'' \models \mathcal{M}$ and $t'\ \mathcal{R}_{\mathcal{C}\cup\mathcal{C}''}\ s'$ where $\mathcal{C}\cup\mathcal{C}'' \in \mathbb{C}^v(Loc)$;

- $t \stackrel{\mathcal{C}'}{\Longrightarrow} t''$ is the result of $n > 0$ τ-transitions. Thus there is t^* such that $t \xrightarrow{(\mathcal{C}^*,\tau)} t^* \stackrel{\mathcal{C}^{**}}{\Longrightarrow} t''$ where $\mathcal{C}^* \cup \mathcal{C}^{**} = \mathcal{C}'$. Hence, $\mathcal{C} \rhd t \xrightarrow{(\mathcal{C}^*\cup\mathcal{C},\tau)} t^* \stackrel{\mathcal{C}^{**}}{\Longrightarrow} t''$. Thus, $\mathcal{C} \rhd t \stackrel{\mathcal{C}'}{\Longrightarrow} t'' \xrightarrow{(\mathcal{C}'',\iota)} t'$ with $\mathcal{C}\cup\mathcal{C}'\cup\mathcal{C}'' \models \mathcal{M}$ and $t'\ \mathcal{R}_{\mathcal{C}\cup\mathcal{C}'\cup\mathcal{C}''}\ s'$ where $\mathcal{C}\cup\mathcal{C}'\cup\mathcal{C}'' \in \mathbb{C}^v(Loc)$.

Now, we show that $(\mathcal{C},\iota).t \sqsubseteq (\mathcal{M},\iota).s \Rightarrow \mathcal{C} \rhd t \sqsubseteq s$. The only transition $(\mathcal{C},\iota).t$ can make is $(\mathcal{C},\iota).t \xrightarrow{(\mathcal{C},\iota)} t$. As $\iota \neq \tau$ and $\iota \in IAct$, according to the third case of the first transfer condition of Definition 7.1, $t\ \mathcal{R}_{\mathcal{C}}\ s$. We construct $\mathcal{R}'_{\{\ \}} = \mathcal{R}_{\mathcal{C}}$ and show that it induces $\mathcal{C} \rhd t \sqsubseteq s$. This is trivial as any transition $\mathcal{C} \rhd t \xrightarrow{(\mathcal{C}\cup\mathcal{C}',\eta)} t'$ is the result of $t \xrightarrow{(\mathcal{C}',\eta)} t'$. The reverse of the rule can be argued in a similar fashion.